2000-03

Full nonlinear simulation of helicopter coupled rotor-fuselage motion using MATLAB Symbolic processor and dynamic simulation

Weissenfels, Robert D.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

FULL NONLINEAR SIMULATION OF HELICOPTER COUPLED ROTOR-FUSELAGE MOTION USING MATLAB® SYMBOLIC PROCESSOR AND DYNAMIC SIMULATION

by

Robert D. Weissenfels

March 2000

Thesis Advisor: E. Roberts Wood
Co-Thesis Advisor: Robert L. King

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2000	Master's Thesis

4. TITLE AND SUBTITLE
Full Nonlinear Simulation of Helicopter Coupled Rotor-Fuselage Motion Using MATLAB® Symbolic Processor and Dynamic Simulation

5. FUNDING NUMBERS
RAA80

6. AUTHOR(S)
Weissenfels, Robert D.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Research Office, Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
37-803

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This thesis formulates the full nonlinear equations of motion for determining the stability of helicopter coupled rotor-fuselage motion utilizing MATLAB®'s Symbolic Math Toolbox. Using the extended symbolic processor toolbox, the goal of this work was to eliminate the time consuming process of converting Fortran or C code generated by the symbolic processor, MAPLE® into a MATLAB® useable format where it is further incorporated into an 'S-function' to be used in the dynamic simulation environment.

The formulation of the equations of motion utilized in this process is unique in that it uses the complete set of nonlinear terms in the equations of motions without utilizing ordering schemes, small angle assumptions, linearizing techniques, or other simplifying assumptions. After derivation, the equations of motion are numerically integrated using the dynamic simulation software SIMULINK® and a time history plot is generated of blade and fuselage motion. The equations of motion are regenerated with each time step allowing the adjustment of characteristic structural, blade and dampening properties. These time traces can be used to explore the effects of damping nonlinearities, structural nonlinearities, active control, individual blade control, and damper failure on ground resonance.

14. SUBJECT TERMS
Nonlinear Simulation, MATLAB® Symbolic Processor, Helicopter Ground Resonance, MATLAB®, SIMULINK®, Numerical Simulation, Mechanical Instability

15. NUMBER OF PAGES
111

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

FULL NONLINEAR SIMULATION OF HELICOPTER COUPLED ROTOR-FUSELAGE MOTION USING MATLAB® SYMBOLIC PROCESSOR AND DYNAMIC SIMULATION

Robert D. Weissenfels
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1990

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2000

Author: [redacted]
Robert D. Weissenfels

Approved by: [redacted]
E. Roberts Wood, Thesis Advisor

[redacted]
Robert L. King, Thesis Co-Advisor

[redacted]
Max F. Platzer, Chairman
Department of Aeronautics and Astronautics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis formulates the full nonlinear equations of motion for determining the stability of helicopter coupled rotor-fuselage motion utilizing MATLAB®'s Symbolic Math Toolbox. Using the extended symbolic processor toolbox, the goal of this work was to eliminate the time consuming process of converting Fortran or C code generated by the symbolic processor, MAPLE® into a MATLAB® useable format where it is further incorporated into an 'S-function' to be used in the dynamic simulation environment.

The formulation of the equations of motion utilized in this process is unique in that it uses the complete set of nonlinear terms in the equations of motions without utilizing ordering schemes, small angle assumptions, linearizing techniques, or other simplifying assumptions. After derivation, the equations of motion are numerically integrated using the dynamic simulation software SIMULINK® and a time history plot is generated of blade and fuselage motion. The equations of motion are regenerated with each time step allowing the adjustment of characteristic structural, blade and dampening properties. These time traces can be used to explore the effects of damping nonlinearities, structural nonlinearities, active control, individual blade control, and damper failure on ground resonance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS AND ACRONYMS

Variables

Phi1, Phi2, Phi3, ...	Azimuth phase angle of each rotor blade
phi	Array of azimuth phase angle components
psi	Azimuth position of each rotor blade with phase angle Phi(n)
Omega	Rotor speed
zeta1, zeta2, zeta, ...	Lead lag displacement of rotor blades
zet	Array of lead lag displacements components
e1	Rotor hinge offset
R	Distance from hinge offset to center of gravity of rotor blade
z	Angle at which lead-lag stops engage
mb1, mb2, mb3, ...	Mass of rotor blades
u1, u2	Fuselage displacements in 1 and 2 direction (x and y)
TBk	Kinetic energy of each rotor blade
rho	Point on elastic axis of rotor blade with respect to the inertial coordinate system at any instant in time
rhoPBd	Position of blade deformed with respect to the undeformed blade
rhoPBd_I	Position of blade deformed with respect to the undeformed blade transformed to inertial coordinate system
rhoBuH	Position of undeformed blade with respect to the hub
rhoBuH_I	Position of the undeformed blade with respect to the hub transformed to inertial coordinate system
rhoHI_I	Position of hub with respect to the inertial coordinate system transformed to the inertial coordinate system
UBk1, UBk2, UBk3, ...	Potential Energy of each rotor blade
Ke1, Ke2, Ke3, ...	Lead-lag linear spring coefficients
Ke	Array of Ke components
Kd1, Kd2, Kd3, ...	Lead-lag nonlinear spring coefficients
Kd	Array of Kd components
Ks1, Ks2, Ks3, ...	Lead-lag stop spring coefficients
Ks	Array of Ks components
DBk1, DBk2, DBk3, ...	Dissapative function of each rotor blade
Czeta1, Czeta2, Czeta3, ...	Lead-lag linear damping coefficients
Vzeta1, Vzeta2, Vzeta3, ...	Lead-lag nonlinear damping coefficients
K1, K2	Effective fuselage stiffness constants (translational)
M1, M2	Effective fuselage mass in 1 and 2 directions(x and y)
c1, c2	Fuselage linear damping coefficients in x and y direction
v1, v2	Fuselage nonlinear damping coefficients in x and y direction
V	Array of nonlinear damping coefficients
TF	Kinetic energy of hub
UF	Potential energy of hub

DF	Dissapative function of hub
F	Array of generalized forces on generalized displacements
F1, F2, F3, F4, F5, ...	Generalized Forces
DOF	Degrees of freedom of system
dDOF	Derivative of DOF with respect to time
ddDOF	Second derivative of DOF with respect to time
DOFq	Substitutive variable for differentiation
dDOFq	Substitutive variable for differentiation
ddDOFq	Substitutive variable for differentiation
DOFB	Degrees of freedom for the blades
DOFF	Degrees of freedom for the fuselage
T	Kinetic Energy
U	Potential Energy
D1	Dissipation function
EOM	Equation of motion
Ax2dot	Second derivative terms
x1dot	First derivative terms

Additional Variables for SIMULINK® S-Function

xXi, xYi	Fuselage states initial displacement conditions
xrXi, xrYi	Fuselage states initial rate conditions
x1i, x2i, x3I	Blade states initial displacement conditions
xr1i, xr2i, xr3i	Blade states initial rate conditions

Additional Variables for Complex (Straub) Model

r1, r2	Fuselage rotation
beta	Flab angular displacement of rotor blade
h	Vertical displacement of hub from fuselage
kf	Flapping spring coefficient of rotor blade
kl	Lead-lag spring coefficient of rotor blade
cf	Flapping damping coefficient of rotor blade
cl	Lead-lag damping coefficient of rotor blade
I11, I22, I12	Coefficients of rotational fuselage motion
KT1, KT2	Translational spring coefficient of fuselage
KR1, KR2	Rotational spring coefficient of fuselage
CT1, CT2	Linear translation damping coefficient of fuselage
CR1, CR2	Linear rotation damping coefficient of fuselage
VT1, VT2	Nonlinear translation damping coefficient of fuselage
VR1, VR2	Nonlinear rotation damping coefficient of fuselage
thet1, thet2, ... , thet7	Pitch angle of section of rotor blade
cd0	Drag coefficient of blade section
mu	Advance ration

lambda	Inflow ratio
a	Length of blade section
c	Chord of blade section
Vair	Relative air velocity
V_I_t	Total velocity in inertial coordinates
V_Bd_t	Total velocity in blade deformed coordinates
UR	Radial component of total velocity
UP	Perpendicular component of total velocity
UT	Tangential component of total velocity
UU	2-D velocity component
aoa	Angle of attack of rotor blade
dFbeta	Blade differential force due to flapping
dFzeta	Blade differential force due to lead-lag
Mbeat_k	Generalized aerodynamic force on blade due to flapping
Mzeta	Generalized aerodynamic force on blade due to lead-lag
GF	Generalized force

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENT

This work has been sponsored by the US Army Research Office, Triangle Park, NC, under contract number 37-803. Their support and interest in the subject matter is greatly appreciated. I would like to thank Professor E. Roberts Wood and Dr. Robert L. King for their guidance throughout this thesis. And I would especially like to thank my wife, Kelly, and our children for their understanding and patience. Without their support, the time and effort required to accomplish this degree would never have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. DISCUSSION

Air and ground resonance are potentially destructive mechanical instabilities that may occur in helicopters due to dynamically coupled interaction between the rigid body motion of the fuselage and oscillations of the rotor blades in their plane of rotation. For over 40 years engineers have felt reasonably comfortable with their ability to predict and control air and ground mechanical instabilities. The self excited vibrations that occur while an aircraft is in contact with the ground is known as ground resonance and this will be the primary focus of this thesis.

Helicopter manufacturers employ several methods of blade root retention to obtain the flapping and lead-lag degrees of freedom necessary for a helicopter rotor blade to operate properly. These methods include: (1) fully articulated; (2) bearingless; (3) and hingeless main rotor designs.

Fully articulated rotor heads utilize two hinges that rotate with and are attached to each rotor blade in order to provide the necessary freedom of movement for the dynamic motion of the rotor blade. One hinge is positioned in a horizontal plane and allows for flapping motion of the rotor blade. The other hinge is positioned in a vertical plane. This hinge allows for lead-lag, movement of the blade in its plane of rotation. The lead-lag hinge generally incorporates some damping. This is generally a hydraulic air-oil strut or an elastomeric design, to constrain the movement of the individual rotor blades. Early designs of these dampers have tended to be heavy and maintenance intensive.

The hingeless/bearingless rotor is a variation of the fully articulated rotor system. In this configuration the designer has substituted a flexible section to replace both the lead-lag hinge as well as the flapping hinge.

In short, by eliminating the need for hinges, improvements in technology, particularly composite materials, have allowed designers to significantly reduce the weight and complexity of modern helicopter rotor systems. However, this reduction/elimination of hinges has also reduced the likely attachment points for linear dampers and other means of controlling the motion of the rotor blades. This further complicates eliminating ground resonance, as well as the associated complexity of manufacturing and maintenance of the dampers. By simplifying and facilitating the process of modeling ground resonance the hope is to open the study of ground resonance to a deeper understanding as well as investigate methods of controlling it. By allowing the variation of parameters and by receiving immediate feedback as to the resultant effects, this program and future versions will help improve the properties of existing damper systems, facilitate the possibility for application of nonlinear dampers, potentially eliminate external dampers using existing material technology, and facilitate the investigation of other aero-mechanical instabilities.

B. BACKGROUND

In the early 1940's the phenomenon of ground resonance was already being investigated by NACA. Robert Coleman's pioneering work was the first definitive analysis to correctly address the issue of ground resonance [Ref. 1] and [Ref. 2]. In his work Coleman identified ground resonance to be a self-excited, mechanical instability

phenomenon. He found the primary modes to be excited in the normal operation of the rotorcraft were the hinged rigid body response of the blades, in their plane of rotation coupled with horizontal deflection of the main rotor pylon.

M. L. Deutsch simplified the results of Coleman and Feingold for the practicing engineer. Based on Coleman's theory he developed a quantitative analysis of the damping required to keep the helicopter free of the instability, often referred to as Deutsch's Criteria [Ref. 3].

Coleman and Feingold's work became the basis for the evolution of theory and design techniques used for dealing with ground resonance. Although the classic theory offers much insight and understanding into the phenomenon, especially for conventional articulated rotor systems, the increasing popularity of hingeless and bearingless rotor designs in modern helicopters called for increasingly more sophisticated analytical tools.

More sophisticated analytical tools became possible with the evolution of digital computers and the improvement in computational power. Peters and Hohenemser [Ref. 24] apply Floquet analysis to the problem of lifting rotor stability. Floquet analysis is a method which can be used to determine the stability of solutions to systems of linear ordinary differential equations with periodic coefficients. The Floquet transition matrix which relates the system state variables at the beginning and end of a rotational period is computed by numerical time wise integration. The eigenvalues of the transition matrix are a measure of system stability. Hammond [Ref. 25] applies Floquet analysis to the prediction of mechanical instabilities, specifically examining the case of unbalanced lead-lag damping. The unbalanced problem requires solution of the equations of motion with

the blade equations expressed in the rotating reference frame because a transformation to the fixed system is no longer possible for a ground resonance analysis as was possible for the isometric case. As a result, you are left with a system of equations with periodic coefficients which can be handled by the Floquet method.

Hingeless and bearingless rotor configurations often face the additional difficulty of air resonance. Aerodynamics may play more than a passive roll in the ground resonance regime in hingeless systems in contrast to articulated systems where aerodynamics have little effect. As a result, more complex models are required to accurately represent the physics of the helicopter aeromechanical stability problem. Models must include blade flap and torsional degrees of freedom as well as lead-lag degrees of freedom. Fuselage models also should include pitch and roll as well as translational degrees of freedom. Aerodynamic models can range from quasi-steady strip theory to unsteady aerodynamic theories which include elaborate wake models or dynamic inflow models. Ormiston [Ref. 26] utilizes a rigid blade and rigid fuselage model with flap-lag and pitch-roll degrees of freedom to conduct parametric investigations based on an eigenvalue analysis. As is typical, the equations of motion were derived by a Newtonian approach and the resulting system of nonlinear differential equations are linearized for small perturbations. The model includes linear rotor blade and landing gear springs, viscous damping, and quasi-steady aerodynamics. Freidmann and Venkatesan [Ref. 27] and Freidmann and Warmbrodt [Ref. 14] derive the complete set of governing equations of a helicopter rotor coupled to a rigid body fuselage. The equations account for rotor blade elastic deformations and include quasi-steady

aerodynamics or modified Theordorsen unsteady aerodynamic theory. In deriving the full equations of motion, Freidmann et al., stress the importance of applying an ordering scheme to the process in order to handle the complexity of the equations and enormous number of terms generated by their expansion. The equations, as presented by Freidmann et al. [Ref. 27 and 14], are in a form which makes them generally applicable to a wide range of rotorcraft problems.

Another interest in the study of helicopter ground resonance is the effect that nonlinear elastic and damping forces have on stability. Tongue [Ref. 23], Tongue and Flowers [Ref. 9 and 8], Tongue and Jankowski [Ref. 28], and Tang and Dowell [Ref. 29], use variations of the nonlinear technique of harmonic balance using describing functions to represent nonlinear damping. The technique is useful for investigating limit cycle behavior of strongly nonlinear systems and its impact on system stability.

Helicopter aeromechanical instabilities can be analyzed by methods ranging from Coleman's classic analysis to direct time integration of the equations of motion. As engineers strive to develop rotor systems free of ground and air resonance which do not require the addition of maintenance intensive mechanical damping systems, more elaborate models will be needed to accurately capture all physical aspects of the problem. To achieve the truly damperless rotor Ormiston [Ref. 30] addresses three different approaches which may be feasible, 1) incorporating high damping material into the blade or flexbeam structure, 2) automatic feedback control, and 3) development of aeroelastic couplings to provide inherent stability. These three approaches provided the impetus behind the work performed by LT Christopher S. Robinson.

A full nonlinear simulation model of coupled rotor/fuselage interaction was created by Robinson in March of 1997 at the Naval Postgraduate School (NPS) [Ref. 4]. This model is a dedicated Coleman analysis tool that initially utilizes a symbolic processor, Maple® to derive the full nonlinear equations of motion of a helicopter using the LaGrange equation. In Robinson's thesis the derived equations of motion are then converted, in Maple®, to Fortran or C and then converted into a MATLAB® programming language. The converted MATLAB® result is incorporated into a SIMULINK® S-function, producing time history plots of blade/fuselage motion. This process is unique in that it uses the complete set of nonlinear terms in the equations of motions without utilizing ordering schemes, small angle assumptions, linearizing techniques, or other simplifying assumptions.

Further development of the NPS modeler was accomplished by Robinson with the help of LCDR Robert L. King, [Ref. 12, 18, 19, 20]. The modeler was used to simulate the Froude Scale RAH-66 [Ref. 21]. It was also used to investigate an interblade coupling approach to improved rotor stability without lag dampers [Ref. 22].

King completed his dissertation in June 1999 [Ref. 5]. In his dissertation, King adopted Robinson's work to further explore the potential of eliminating the snubber-damper or damper on hingeless rotor designs and replace it with a flexbeam that has been modified to possess nonlinear properties. It is King's research in this field that has provided the motivation to accomplish the following work.

Since Robinson's original work, MATLAB® has incorporated a symbolic processing toolbox into its software. By using the extended symbolic processor toolbox

in MATLAB® the goal is to eliminate the awkward and time consuming process of converting the Fortran or C code generated by MAPLE® into a MATLAB® useable format where it is further incorporated into an 'S-function' to be used in the dynamic simulation environment. Direct interface between the derivation of the equations of motion and the actual simulation process results in fewer steps required to develop a system, it reduces the computer programs required to run a simulation, and reduces the potential contamination of the solution that may result from human interaction. Hopefully this will encourage further use of the rotor-fuselage coupled motion simulation and enable researchers of all levels to explore the phenomenon of ground resonance.

After a brief explanation of the work performed by Robinson, Rafanello, and King an introduction to MATLAB®, MATLAB® the symbolic processing toolbox, the S-function, and the Simulink model will be provided. The goal, again, is to provide a path for future use of the NPS modeler.

THIS PAGE INTENTIONALLY LEFT BLANK

II. CURRENT MODEL

A. REVIEW OF WORK PERFORMED BY ROBINSON

1. Simple Model

As with this thesis, Robinson's work began with a simplified model using reduced degrees of freedom on a three bladed model. The simplified model is based on that of Coleman as is shown in Figure 1, [Ref. 1]. Elastic forces generated by rotor blade and flexbeam motion were modeled as a linear torsional spring located at the effective hinge position of the blade. Landing gear stiffness was also modeled with linear springs. The landing gear and lead-lag dampers were modeled with linear dashpot type dampers. Transformation between the various systems, in order to develop Lagrange's equations of motion, were accomplished using Euler angle rotations. For the simple model the coordinate transformations used were hub (parallel to fuselage but offset a distance h in the positive z direction) to inertial (fixed relative to the earth), blade undeformed (fixed to the effective hinge position) to hub, and blade deformed (fixed to the effective hinge position with the x-axis coincident with the blade 'elastic' axis) to blade undeformed.

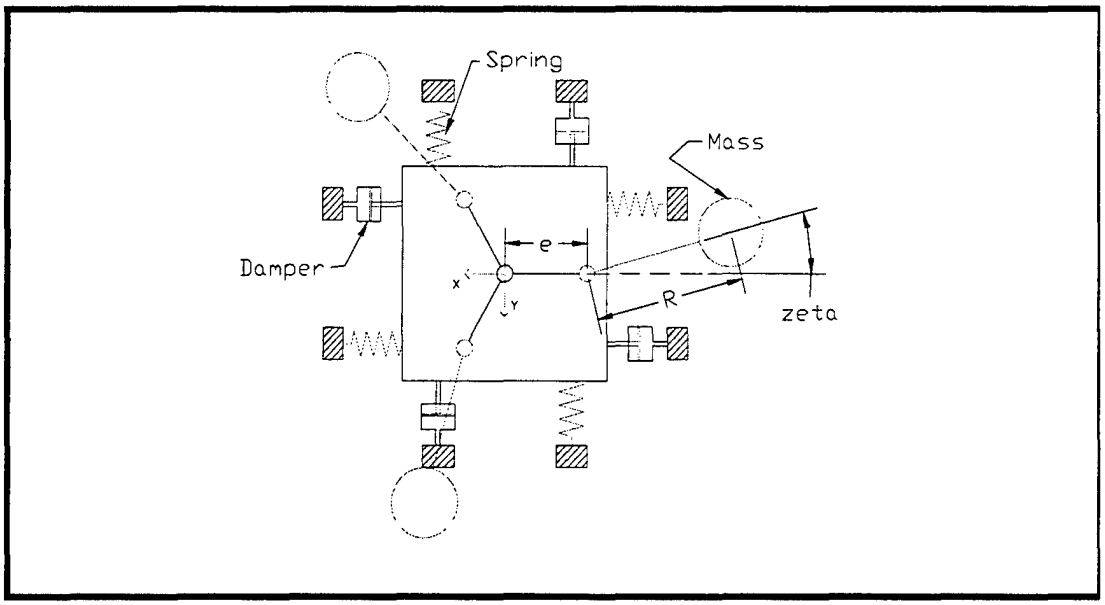

Figure 1. Simplified Rotor Model [From Ref. 4]

Robinson showed excellent agreement between his model and the Coleman model, with departure occurring only when displacements are very large, Figure 2. This is to be expected since Robinson's model does not assume small angle theory whereas the Coleman-Feingold-Bramwell model does.

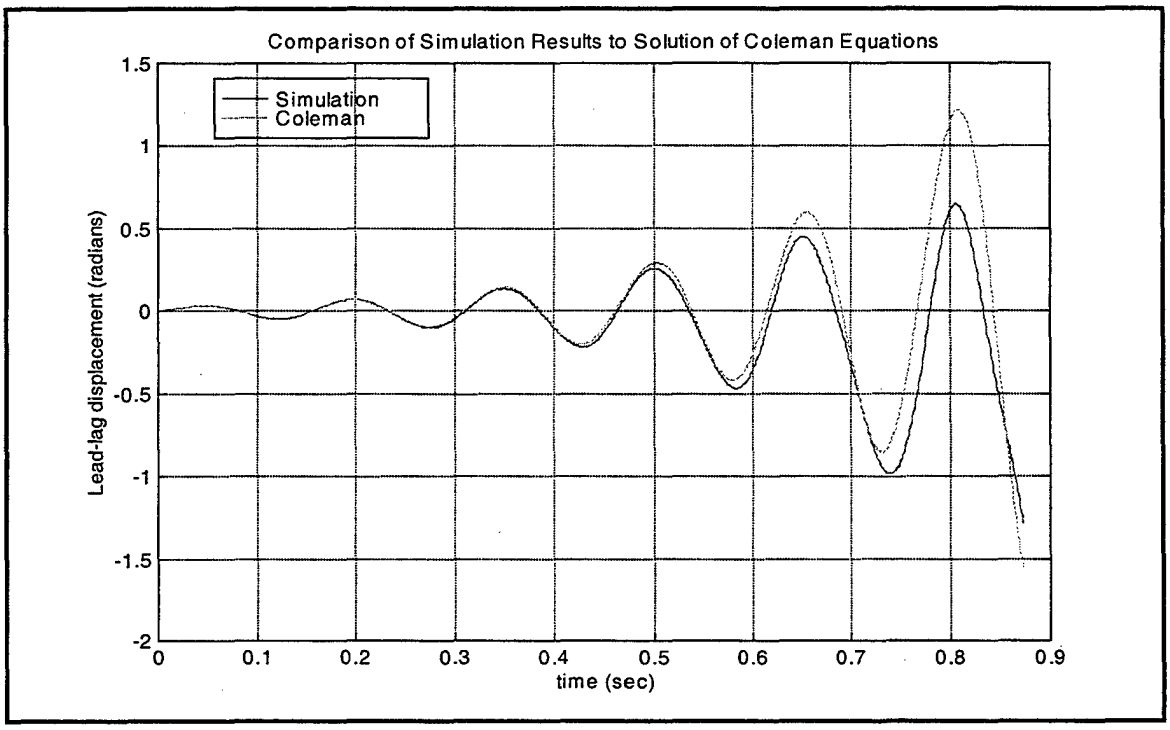

Figure 2. Validation of model with solution of Coleman's model [From Ref. 4]

Figure 3 is a parametric plot of the simple model with an isotropic pylon and rotor. The damping ratio from the moving block result is plotted versus ω/ω_f for various Deutsch numbers [Ref. 15].

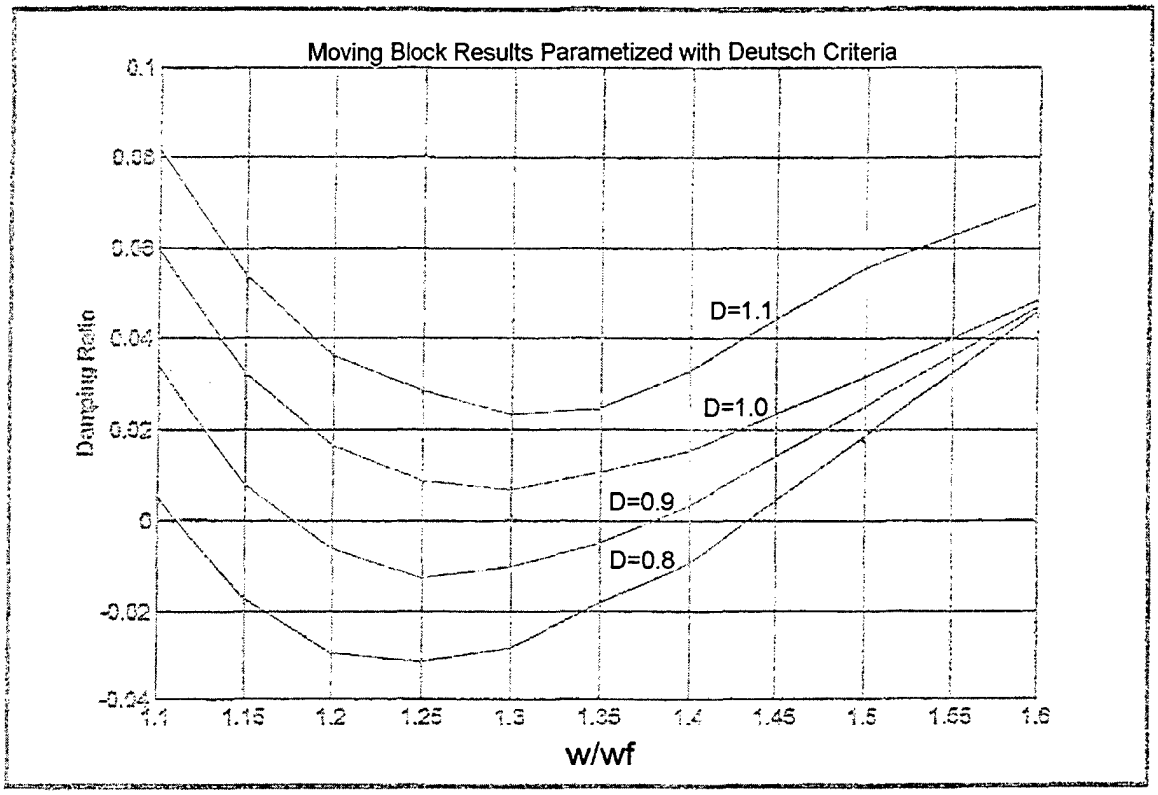

Figure 3. Validation of model against Deutsch Criteria [After Ref. 4]

This model allows for the following degrees of freedom:

u1 - Fuselage translation in 1-direction (x-direction)

u2 - Fuselage translation in the 2-direction (y-direction)

ζ_K - Lead-lag angular displacement of k^{th} rotor blade

2. Complex Model

The complex model is based on that used by Straub [Ref. 6]. This model assumes rigid blades and fuselage, as in the simple model, with flap and lead-lag hinges as coincident. It incorporates fuselage rotation as well as blade flap, requiring additional

transformations. Figures 4 and 5 show a schematic representation of the complex model and the transformations utilized in the derivation. The transformations required in the complex model are fuselage (fixed to center of gravity of fuselage) to inertial, hub to fuselage, blade undeformed to hub, blade deformed to blade undeformed.

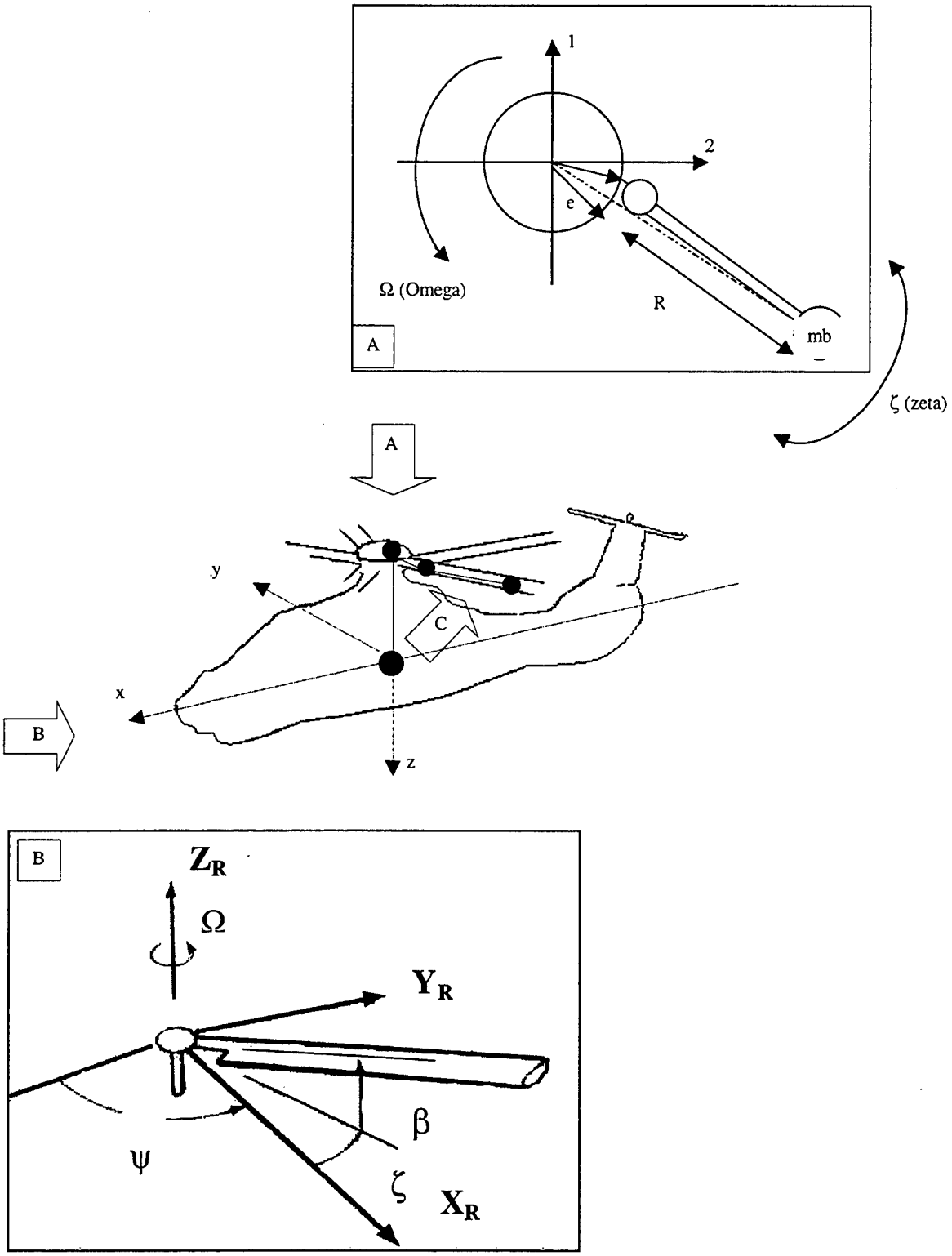

Figure 4. Schematic representation of complex model transformations

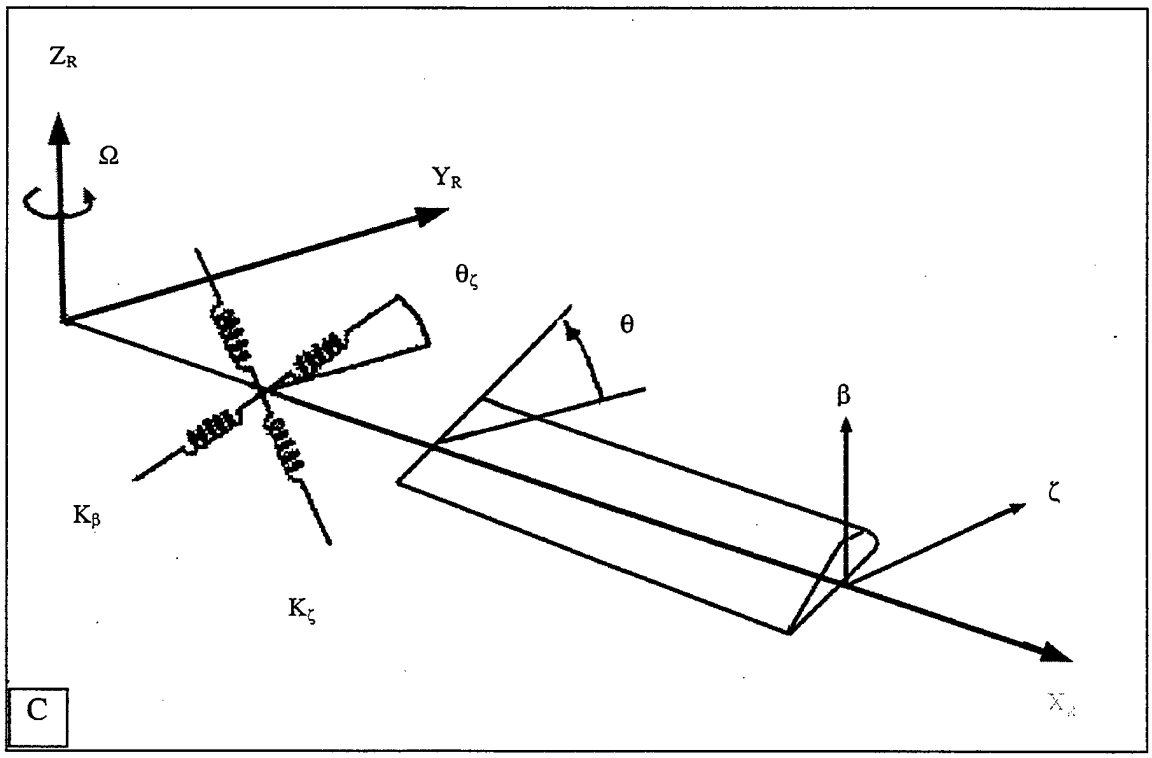

Figure 5. Schematic representation of complex model transformations

This model utilizes the following degrees of freedom:

u1 - Fuselage translation in 1-direction (x-direction)

u2 - Fuselage translation in the 2-direction (y-direction)

r1 – Fuselage rotation about 1-axis (roll)

r2 – Fuselage rotation about 2-axis (pitch)

ζ_κ - Lead-lag angular displacement of k^{th} rotor blade

β_κ - Flap angular displacement of k^{th} rotor blade

3. Derivation of the Lagrange Equation

A brief overview of the derivation accomplished by Robinson will be included in this thesis. For a more extensive explanation see Robinson's thesis [Ref. 4]. The Lagrangian equation may be expressed as follows:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right)-\frac{\partial T}{\partial q}+\frac{\partial U}{\partial q_i}+\frac{\partial D}{\partial \dot{q}_i}=F_i$$

Where T is the kinetic energy, U is the potential energy, D is the dissipation function, F_i is the generalized force, and q_i is the generalized displacement. In the complex model the generalized force term F_i, will describe the aerodynamic forces on the individual rotor blades. This derivation develops a system of homogeneous equations. The various energy terms are broken down into two categories, blade motion and fuselage motion to give the following equations:

$$T = T_F + \sum_{k=1}^{N} (T_B)_k$$

$$U = U_F + \sum_{k=1}^{N} (U_B)_k$$

$$D = D_F + \sum_{k=1}^{N} (U_B)_k$$

Where the subscripts F and B indicate the fuselage and rotor blade respectively.

a. Kinetic Energy Terms

The kinetic energy of the kth rotor blade is given by the following expression:

$$(T_B) = \int_0^R \frac{1}{2} m'(\dot{\vec{\rho}} \cdot \dot{\vec{\rho}}) dr$$

Here $\vec{\rho}$ is the position of a point on the elastic axis of the k[th] rotor blade with respect to the inertial coordinate system at any instant in time, and m' is the mass distribution per unit length of the blade (mass distribution per unit length is assumed to be uniform). The position of a point on the elastic axis of a rotor blade, $\vec{\rho}$, is expressed as the sum of relative positions with respect to the various coordinate systems transformed to the inertial system. Thus,

$$\vec{\rho} = (\vec{\rho}_{F\_I})_I + (\vec{\rho}_{H\_F})_I + (\vec{\rho}_{Bu\_H})_I + (\vec{\rho}_{Bd\_Bu})_I + (\vec{\rho}_{P\_BD})_I ,$$

where, for example, the term $(\vec{\rho}_{Bu\_H})_I$ is the position of the origin of the undeformed blade coordinate system with respect to the hub coordinate system transformed into the inertial coordinate system.

Following the proper transformations, the resultant expression provides the position of an arbitrary point on the elastic axis of the k[th] rotor blade, with respect to the inertial coordinate system, at any instant in time, in terms of the system degrees of freedom. The time derivative of this expression gives the velocity, $\dot{\vec{\rho}}$, which is substituted into the equation for kinetic energy for the k[th] rotor blade.

The fuselage kinetic energy in terms of translational and rotational degrees of freedom is

$$(T_F)_{trans} = \frac{1}{2} M_1 \dot{u}_1^2 + \frac{1}{2} M_2 \dot{u}_2^2$$

$$(T_F)_{rot} = \frac{1}{2} I_{11} \dot{r}_1^2 + \frac{1}{2} I_{22} \dot{r}_2^2 - 2 I_{12} \dot{r}_1 \dot{r}_2$$

b. Potential Energy Terms

The elastic forces generated by rotor blade motion give rise to a potential energy term in the Lagrange equation. Since a rigid blade model was assumed, the potential energy was modeled using equivalent torsional springs to restrain the rotor

blade, with spring constants selected to approximate elastic forces due to in plane and out of plane bending of the rotor blade (and the flexbeam in the hingeless case). The potential energy of the kth rotor blade is

$$(U_B)_k = \frac{1}{2} K_B \beta_k^2 + \frac{1}{2} K_\varsigma \varsigma_k^2$$

The fuselage potential energy in terms of translational and rotation degrees of freedom is

$$(U_F)_{trans} = \frac{1}{2} K_1 u_1^2 + \frac{1}{2} K_2 u_2^2$$

$$(U_F)_{rot} = \frac{1}{2} K R_1 r_1^2 + \frac{1}{2} K R_2 r_2^2$$

An explanation of the validity of using an equivalent torsional spring system to model the elastic forces of a deformed rotor blade is given in some detail in Venkatesan and Friedmann [Ref. 7].

c. Dissipation Function Terms

System damping is modeled in energy form by use of a dissipation function, which for the kth rotor blade of he complex rotor model is

$$(D_B)_k = \frac{1}{2} C_\beta \dot{\beta}_k^2 + \frac{1}{2} C_\varsigma \dot{\varsigma}_k^2$$

The dissipation function for the fuselage in terms of translational and rotational degrees of freedom is

$$(D_F)_{trans} = \frac{1}{2} C_1 \dot{u}_1^2 + \frac{1}{2} C_2 \dot{u}_2^2$$

$$(D_F)_{rot} = \frac{1}{2} C\ R_1 \dot{r}_1^2 + \frac{1}{2} C\ R_2 \dot{r}_2^2$$

d. Resultant Equations

It is important to note that when applying Lagrange's equation in MAPLE®, derivatives with respect to the degrees of freedom and the time rates of change

of the degrees of freedom, the time functional notation which represents these variables must be converted to independent variable notation. For example, the flap angle degree of freedom, $\beta(t)$, and its time rate of change, $\frac{\partial \beta(t)}{\partial t}$, would have to be replaced in all of the energy expressions by the independent variables, β and $d\beta$ respectively, in order for terms such as $\frac{\partial T}{\partial q_i}$ and $\frac{\partial T}{\partial \dot{q}_i}$ (where $q_i = \beta(t)$ and $\dot{q}_i = \frac{\partial \beta(t)}{\partial t}$) to be evaluated properly by the MAPLE® symbolic engine. Additionally, for the time derivative term, $\frac{\partial}{\partial t}\left(\frac{\partial T}{\partial \dot{q}_i}\right)$, to be evaluated properly, all degrees of freedom expressed in independent notation must be converted back to time dependent notation.

The equations of motion generated for the simplified model (Coleman model) in this format were compared, by Robinson, to the equations used by Flowers and Tongue [Ref. 8,9]. The equations were found to match exactly except for the nonlinear damping terms that were not included initially but are later included in the simulation. Robinson's work has been further utilized by LT Salvator Rafanello and LCDR Robert L. King.

B. REVIEW OF WORK PERFORMED BY RAFANELLO

Rafanello's work further validated the NPS ground resonance modeler by matching simulation results with Professor Roberts E. Wood's HSS-2 ground resonance analysis [Ref. 10, 11]. Rafanello used a five bladed version of the simplified model adapted to the H-3 Sea King. A mass-spring-damper system of the H-3's landing gear

was examined to obtain parameters to enter into the NPS simulation. A thorough analysis of the natural frequencies in the lateral and roll modes at three power settings of 0/20/80 percent airborne were made with respect to the literature prepared by Coleman, Feingold, and Deutsch. Stability charts were developed from the tailored data of the Sea King and compared with the modeler's results. Through these results Rafanello was able to directly link the classical work with the NPS modeler in the roll and lateral modes of the H-3.

In his work, Rafanello was also able to verify the conservative nature of Deutsch's Criteria as was presented in Robinson's thesis. He was able to determine that Deutsch's Criteria is conservative in that the critical point for his criteria, or bucket of each curve, still shows positive damping when analyzed with the NPS simulation. Please refer to Reference 10 for further analysis.

C. REVIEW OF WORK PERFORMED BY KING

King's work focused on examining alternative designs for a damperless helicopter rotor. He explored the potential of eliminating the snubber-damper or damper on hingeless rotor designs and replacing it with a flexbeam that is modified to possess nonlinear properties. The NPS ground resonance modeler is well suited to this task in that it can accurately model nonlinear mechanical properties. King included nonlinear properties in the blade root to produce potentially acceptable bounded responses in the parameter region, with the NPS modeler, where linear theory would have predicted instability. He was also able to determine regions of unacceptable response where linear theory would have predicted stability.

King also looked at linearly linked rotor systems and unevenly spaced rotors using the NPS modeler. By linearly linking the rotors he was able to change the mode shapes of the rotor blades' lead/lag motion. This alters the blade natural frequency in the lead/lag and regressing mode of the rotor. Therefore, by linking the blades, the potential exists to avoid ground/air resonance by detuning the rotor-body dynamics. This method avoids ground and air resonance in a similar manner to the nonlinear stiffness approach. Unevenly spacing the rotor blades produced the same mechanical instabilities as were produced without altering the rotor configuration, for all cases tested.

King was further able to use the NPS modeler to simulate the performance of a 1/6 scale Comanche rotor system. The goal of the simulation was to match the fixed system damping values to the test data and to UMARC (computer simulation modeler designed at University of Maryland). In this comparison the NPS modeler shows slightly better results than UMARC at the lower RPM's tested. At higher rotor speed, the difference between the two simulations is negligible, Figure 6.

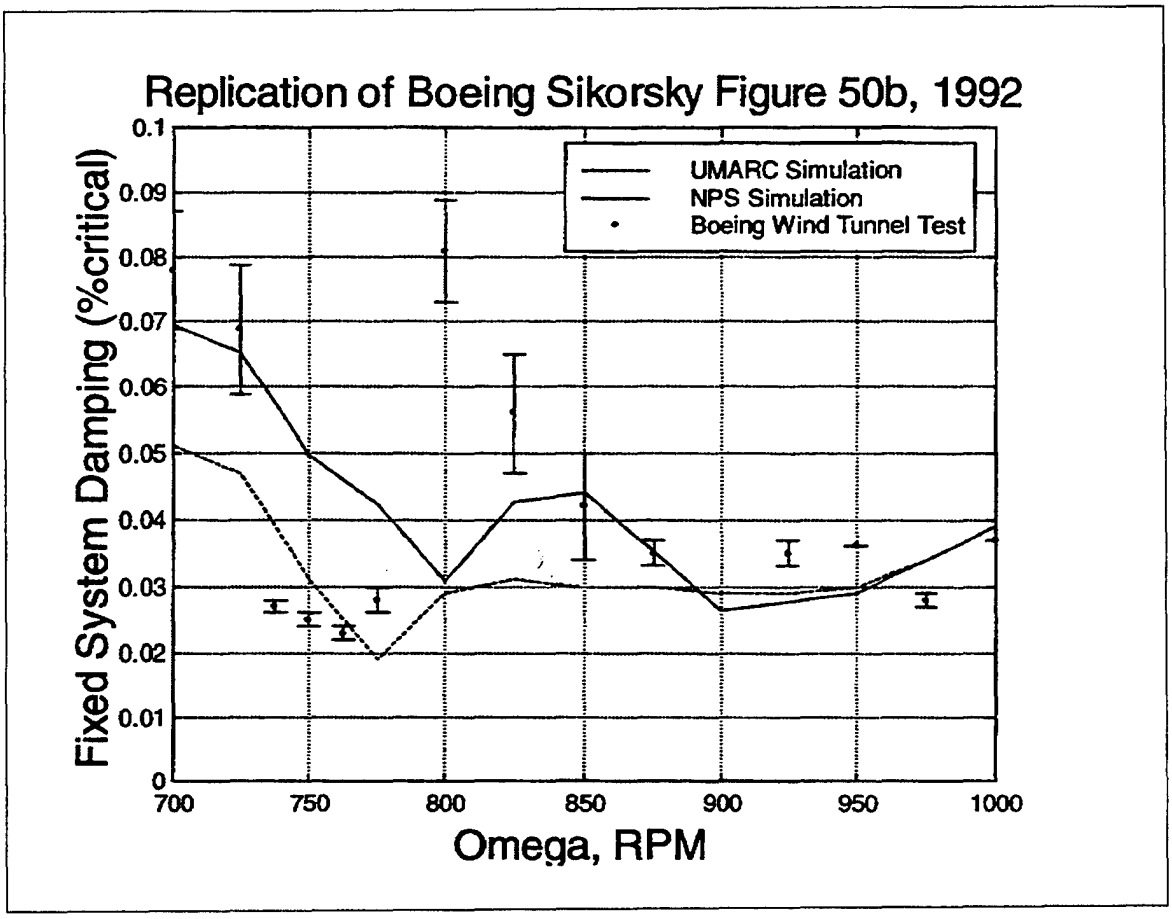

Figure 6. Final NPS simulation results with increased geometric gain [From Ref. 5]

From these results King concluded that "the NPS modeler shows potential in modeling rotors with nonlinear physical parameters and further development should be considered" [Ref. 5].

III. MATLAB®

A. INTRODUCTION AND HELP

The NPS modeler has been shown to be effective in modeling coupled rotor-fuselage motion. In order to help facilitate the functionality of the simulation program, the inclusion of a symbolic processing toolbox in MATLAB provides a means of refining the simulation process. The computer programming contained within this thesis is accomplished entirely in the computer programs MATLAB® and SIMULINK®. Both programs are developed by Mathworks, Inc. and are run interchangeably. MATLAB® is an array and matrix based program allowing programming features similar to other computer programming languages. In addition to its matrix orientation, MATLAB® also offers Graphical User Interface (GUI) tools and an excellent graphics package. MATLAB® stores all data as arrays, which lends itself to the manipulation of sets of data in a wide variety of ways. Recently Mathworks, Inc. has incorporated the Symbolic Math Toolboxes into MATLAB®'s numeric environment. These toolboxes supplement MATLAB®'s numeric and graphical facilities with several other types of mathematical computation:

Facility	Covers
Calculus	Differentiation, integration, limits, summation, and Taylor series
Linear Algebra	Inverses, determinants, eigenvalues, singular value decomposition, and canonical forms of symbolic matrices
Simplification	Methods of simplifying algebraic expressions
Solution of Equations	Symbolic and numerical solutions to algebraic and differential equations
Solution of Equations	Symbolic and numerical solutions to algebraic and differential equations
Variable-Precision	Numerical evaluation of mathematical expressions to any

Arithmetic	specified accuracy
Transforms	Fourier, Laplace, z-transform, and corresponding inverse transforms
Special Mathematical Functions	Special functions of classical applied mathematics

Table 1. MATLAB® Symbolic Math Toolboxes [After Ref. 13]

The computational engine underlying the toolboxes is the kernel of MAPLE®, a system developed primarily at the University of Waterloo, Canada, and, more recently, at the Eidgenössiche Technische Hochschule, Zürich, Switzerland. MAPLE is marketed and supported by Waterloo Maple, Inc.

Maple V Release 4 was incorporated into MATLAB® 5. There are tow toolboxes. The basic Symbolic Math Toolbox is a collection of more than one hundred MATLAB® functions that provide access to the Maple kernel using a syntax and style that is a natural extension of the MATLAB® language. The basic toolbox also allows you to access functions in MAPLE®'s linear algebra package. The Extended Symbolic Math Toolbox augments this functionality to include access to all nongraphics MAPLE® packages, MAPLE® programming features, and user defined procedures. With both toolboxes, you can write your own M-files to access MAPLE® functions and the MAPLE® workspace [Ref. 13].

In order to determine it you have the Extended Symbolic Math Toolbox a simple check on the MATLAB® command line will work. The syntax is as follows:

```
>>maple('with','numtheory')
```

This will either:

1) list a set of libraries that are loaded or
2) produce the following error:

```
??? 'with'(...)' requires extended symbolic toolbox.
Error in ==> Paul P.:Applications :MATLAB 4.2a:Toolbox:Symbolic
```

```
Installer:symbolic:maplemex.mex
Error in ==> Paul P.:Applications :MATLAB 4.2a:Toolbox:Symbolic
Installer:symbolic:maple.m
On line 84  ==> [result,status] = maplemex(statement);
```

More information about obtaining the Extended Symbolic Math Toolbox may be obtained from the Mathworks website [www.Mathworks.com].

This thesis is written assuming the reader possesses a basic understanding of MATLAB®. Additional help with the Symbolic Math Toolbox may be obtained in several ways. General information about symbolic functions may be obtained by typing,

>> help sym/*function*.

Here, function, is the name of an "overloaded" MATLAB® numeric function. This method provides symbolic-specific implementation of the function, using the same function name. This provides some consistency in the logically programming process and the functions used in both MATLAB® and MAPLE®. For example *diff* is a function that is used numerically as well as symbolically by MATLAB®.

>> help *diff*

will produce a different result that

>> help sym/*diff*

Help for the numeric version informs you that the function name is "overloaded". Overloaded methods

>> help char/diff.m

>> help sym/diff.m

Help may also be obtained on MAPLE® commands from the MATLAB® work space.

```
>> mhelp diff
```

This returns the help page for the MAPLE® diff function.

B. IDENTIFYING SYMBOLIC VARIABLES IN MATLAB®

Using the Symbolic Math processing capabilities of MATLAB® is different than using MAPLE®. Of particular difference is the requirement that a variable be identified as a symbolic variable before it used in MATLAB®. If you have ever received the error

```
???Undefined function or variable 'r'
```

you have fallen victim to this peculiarity. The exception to this rule is if a variable is the result of an equation consisting of previously identified symbolic variable. Then the variable is automatically assumed to by symbolic.

A symbolic variable may be identified in at least two methods. To designate one variable as a symbolic expression use

```
u = sym('u')
```

or

```
b = sym('beta').
```

Here 'u' is established as a symbolic variable u and b as beta. These variables are defaulted to be with respect to x. A variable my also be assigned to an expression,

```
f = sym('a*x^2 + b*x  + c')
```

but in this form the Symbolic Math Toolbox does not create variables corresponding to the terms of the expression, a, b, c, x. To perform symbolic math operations (e.g., integration, differentiation, substitution, etc.) on f, each variable must be created

explicitly. This may be accomplished in the form above or by typing,

```
sym a b c x.
```

Again these variables are defaulted to be with respect to x. The independent variables is chosen with the idea that they are typically lower-case letters found at the end of the Latin alphabet (e.g., x, y, or z). Therefore the closest letter to 'x' alphabetically is chosen as the default symbolic variable. If there are two equally close, the letter later in the alphabet is chosen. Establishing a variable with respect to another variable may be accomplished in the following format,

```
zet = sym('zet(t)').
```

The variable zet is now defined with respect to 't'. This is of particular importance when performing integration and differentiation as in this thesis. Further explanations of defining symbolic variables in MATLAB® may be found in reference 13 symbolic toolbox.

C. SUBSTITUTIONS

When processing expressions using the Symbolic Math Toolbox, the MAPLE® kernel, accessed by MATLAB®, does not "a priori", treat the expression as a number. MAPLE® assumes that the symbolic variables as "a priori" indeterminate. That is, they are purely formal variables with no mathematical properties. Consequently, when calculating an expression, the variables do not automatically assume a numerical value, as assigned. For example,

```
>> syms x y z
>> l = x + y * z
>> x = 1;
```

```
>> y = 3;
>> z = 5;
>> l
```

produces the answer,

```
l = x^2+ y * z
```

However,

```
l = subs(l)
```

produces

```
16
```

In many cases a specification of the desired form of output is required. If a numerical output is desired for the expression and it requires processing beyond basic arithmetic, the 'subs' function may be incorporated with the 'double' command to produce a double precision array. Double precision array is the general MATLAB® workspace parameter. Inquiry into the class of each variable may be accomplished by typing,

```
>> whos.
```

A list of the Name, Size, Bytes, and Class of all variable accessible to the workspace are listed.

D. MAPLE® FUNCTION

The 'maple function' allows the user to access MAPLE® functions directly from the MATLAB® workspace. This function takes sym objects, strings, and doubles as inputs and returns a symbolic object, character string, or double corresponding to the class of the input.

It is important to know the syntax of the calling sequence used by MAPLE® function to be used. This information may be obtained by using 'mhelp.' The general format for using the MAPLE® function is,

```
A(i,j) = maple('coeff',EOM(i),ddDOFq(j)),
```

where 'coeff' is the maple function to be used. In this case, 'coeff' sets A(i,j) equal to the ddDOF(j) coefficients of the expression EOM(i). In some instances the format of the MAPLE® function is not consistent with form used in MAPLE®. For example the MAPLE® function 'union' requires the sets to be evaluated to be on either side of the function instead of in parentheses after it:

set1 := setA union setB

instead of,

set1 = union(setA,setB).

These forms are not conducive to using the maple function however the proper form by be accomplished by using strings and converting the syntax. An example of this is provided in Appendix I with the MATLAB® function munion.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MAPLE® TO MATLAB® CONVERSION

A brief explanation of the programming tools required to produce the results of this thesis is provided above. Unfortunately, excessive trial and error went into establishing the differences in the programming format required to produce the desired output. The similarities in the two languages proved to be a hindrance rather than beneficial.

A. DEFINING THE VARIABLES

In order to accomplish the derivation of the equation of motion using Lagrange's equation in MATLAB® the following hurdles need to be crossed. MAPLE® allows variables to be subindexed i.e.

. $\psi = \Omega t + \phi_k$

where ϕ_k is subindexed with k, as is commonly used in mathematical text books. In this case k represent an individual rotor blade and for the three bladed model k = three. This mathematical formulation is represented as

```
psi:=Omega*t+Phi[k];
```

in MAPLE®. MATLAB® uses square brackets, [ ], to represent an array or matrix. As a result there is no way to represent k in the MATLAB® workspace as it is used in MAPLE®. A multidimensional array was used to resolve this conflict. A third index is used to represent the values of each blade. In this form MATLAB® calculates 'psi' three time for a three bladed model. The syntax in this form is,

```
Phi = [Phi1 Phi2 Phi3];
```

Psi = Omega*t + Phi;

But when the variable is called, it is done so with a for loop from 1 to k (k represents the number of blades).

```
for i  = 1:k

m1(:,:,i) = [cos(psi(i) sin(psi(i)) 0
    -sin(psi(i)) cos(psi(i)) 0
    0 0 1];

end;
```

m1 is generated 3 times with a different psi each time. This results is psi with Phi, Phi2, and Phi3 in each successive generation. This accomplishes the necessary 'place holding' required for each blade's energy terms to accomplish the derivation.

B. SET MANIPULATION

As a result of MAPLE®'s dependence on taking derivatives with respect to independent variables, the time function notation, which represents the degrees of freedom and the time rates of change of the degrees of freedom, must be converted to independent variable notation as noted in reference 4. To accomplish this a substitution of sets was used to switch the necessary terms before and after the differentiation. In MATLAB® this may be accomplished in two ways. The first is similar to the form used in the original derivation.

```
DOFF = [u1,u2];
DOFB = [zet1, zet2, zet3];
DOF = [DOFF, DOFB];
dDOF = diff(DOF,t);
ddDOF = diff(dDOF,t);

syms q1 q2 q3 q4 q5 dq1 dq2 dq3 dq4 dq5 ddq1 ddq2 ddq3
```

```
    ddq4 ddq5

DOFq = [ q1 q2 q3 q4 q5];
dDOFq = [ dq1 dq2 dq3 dq4 dq5];
ddDOFq = [ ddq1 ddq2 ddq3 ddq4 ddq5];
set1 = [];
set2 = [];

for i=1:5 % size of DOFq vector
   set1 = [set1 maple('='',DOF(i),DOFq(i)) ];
   set1 = [set1 maple('='',dDOF(i),dDOFq(i)) ];
   set1 = [set1 maple('='',ddDOF(i),ddDOFq(i)) ];
   set2 = [set2 maple('='',DOFq(i),DOF(i)) ];
   set2 = [set2 maple('='',dDOFq(i),dDOF(i)) ];
   set2 = [set2 maple('='',ddDOFq(i),ddDOF(i)) ];
end
set1 = maple('convert',set1,'set');
set2 = maple('convert',set2,'set');
```

This method defines the terms required as symbolic variables. It defines the sets and then uses a maple function to set the terms equal to each other for later substitution. It then uses another maple function to convert the sets into a form that is recognizable by MATLAB®.

The second method uses the 'subs' command local to the MATLAB® workspace. By establishing the necessary variables into an array of 'old' and 'new' terms they can be switched term for term in a given variable (array).

```
DOFF = [u1,u2];
DOFB = [zet1, zet2, zet3];
DOF = [DOFF, DOFB];
dDOF = diff(DOF,t);
ddDOF = diff(dDOF,t);
tempDOF = [DOF dDOF ddDOF];

syms q1 q2 q3 q4 q5 dq1 dq2 dq3 dq4 dq5 ddq1 ddq2 ddq3
     ddq4 ddq5

DOFq = [ q1 q2 q3 q4 q5];
```

```
dDOFq = [ dq1 dq2 dq3 dq4 dq5];
ddDOFq = [ ddq1 ddq2 ddq3 ddq4 ddq5];
tempDOFq = [DOFq dDOFq ddDOFq];

Temp = subs(T,dDOF,dDOFq);
```

T, u1, u2, zet1, zet2, zet3 are predefined arrays and tempDOFq is used in a later portion of the derivation. The speed required to run each method varies depending on the operation to be accomplished. For a simple substitution, the MATLAB® 'subs' function proved to be the quicker method but when larger symbolic terms were involved the first method proved to be the quicker solution. Therefore an optimization was done to determine the optimum combination of the two methods in order to minimize the computational time required to complete the simulation.

C. STATE DERIVATIVE CALCULATION

Using the Symbolic Math Toolbox in MATLAB allows calculation of the state derivatives without first creating a 'B' array, as is necessary with the MAPLE® version. The B array is used to convert the output to C or Fortran code, before it is placed in a MATLAB® function. By avoiding this step, both the MATLAB® function and the SIMULINK® function can call the state derivative directly from the workspace. This is extremely beneficial in that it eliminates any errors associated with converting generated code to a format that usable in SIMULINK®.. It eliminates extra steps in the setup process, helps facilitate simulations, and reduces the time taken to convert the generated code.

D. VARIABLE ROTOR BLADE MODELING

After the code was found to run successfully in SIMULINK®, it was further adjusted to allow for a variation in the number of rotor blades. In the MAPLE® version this was self-contained by using the subindex 'k'. In the MATLAB® version, without the capability of subindexing, each array was built up as a temporary array, capable of deriving a seven bladed model, then only the necessary terms were used to complete the derivation.

E. COMPLEX MODEL

The complex model as derived by Robinson, [Ref. 4], was converted into a MATLAB® function, Appendix E. This model is very applicable to future helicopter design due to the added degrees of freedom. It is the complex model and variations of it that will allow the designer to examine the stability characteristics of rotor systems without expensive and time consuming tests in a wind tunnel. The flexibility in alteing the degrees of freedom and accessing the equations of motion with each time step are what make the simulation useful beyond the calculations performed using the simple model. Additionally the inclusion of nonlinear terms opens a whole new field of dynamic modeling to be considered. Some of the possible areas of exploration are damper springs, duffing springs, and the stability characteristics of varying the length rotor blades.

F. MAPLE® PROCEDURE

Reference 13 describes a method to input a MAPLE® symbolic derivation into the MATLAB® workspace. Creating a source file using a 'maple procedure' the

Extended Symbolic Math Toolbox is capable of reading the source file. Unfortunately the results produced little to no gain due to the process of changing the source code identifier. This process was further complicated due to windows automatically associating the file as a text file or attaching its own identifier to file. Accessing the sources code was eventually possible by changing the file in DOS mode. This process could possibly be simplified using a UNIX based system however it does not produce the ultimately desired results of this thesis.

V. SIMULATION MODEL

A. ORIGINAL SIMULINK® S-FUNCTION

Introduction of the rotor-fuselage dynamics into the Simulink® simulation is accomplished through the use of a Simulink® S-function. The S-function is a user defined Simulink® block that outputs the rotor time histories as functions of the user defined rotor and fuselage parameters. SIMULINK® is a convenient Graphical User Interface (GUI) that performs simulation time integration, allows user input changes between simulations, and provides the user with graphical output.

SIMULINK® accesses an S-function through is numerical integration routines. The routines make calls to the S-function for specific information, the type of information returned is dependent on the value of a flag variable sent by the integration routine. For example,

flag = 0 S-function returns sizes of parameters and initial conditions,

flag = 1 S-function returns state derivatives dx/dt,

flag = 3 S-function returns outputs.

The section of the S-function which computes the derivatives at each time step is the section which calls the equations of motion.

The Lagrangian derivation of the equations of motion generated the equations in the form,

$$\vec{F}(\ddot{\vec{x}}, \dot{\vec{x}}, \vec{x}, t) = 0$$

where $\vec{x}$ is a vector of displacement degrees of freedom of the system. The output of the derivation was further manipulated into the equivalent form,

$$[A(\dot{\vec{x}}, \vec{x}, t)]\ddot{\vec{x}} = \vec{f}(\dot{\vec{x}}, \vec{x}, t)$$

where A is an NxN matrix and f is an Nx1 vector, with N = number of degrees of freedom of the system. This is possible since the equations are quasi-linear in the second derivative terms, i.e., no terms of types such as $\ddot{\vec{x}}$, or $\sin(\ddot{\vec{x}})$, etc. This form can then be transformed from N second order equations to 2N first order equations as follows,

$$\dot{\vec{x}} = \vec{w}$$

$$\dot{\vec{w}} = [A]^{-1}\vec{f}$$

These equations are evaluated at each time step in a numerical simulation to give the state derivatives. This study uses a Runge-Kutta algorithm in SIMULINK® to solve the numerical ordinary differential equations. The Runge-Kutta algorithm generally outperforms other schemes for systems of nonlinear ordinary differential equations which are not too stiff and they handle discontinuities well [Ref. 14].

A Coleman-like 3-bladed hub-rotor model was used for the initial simulation model. It provided for linear stiffness and damping, quadratic damping, and cubic stiffness in the blade degrees of freedom. Other, simulations for damperless rotor analysis have been created for four-blade and five-blade hub-rotor simulations. Table 2 provides a break down of the variables used in the S-function and Figure 7 shows the SIMULINK® model utilized for the simple model simulation.

PARAMETER	PARAMETER SETTINGS	UNITS
Rotor Blade Mass	mb(1) mb(2) mb(3)	mass
Fuselage effective in x and y direction	M(1) M(2)	mass
Distance from hinge to center of mass of blade	R	length
Rotor Speed	Omega	rad/sec
Hinge Offset	e1	length
Angle at which lead-lag stops engage	z	radians
Azimuth phase angle of rotor blade	Phi(1) Phi(2) Phi(3)	radians
Lead-lag linear damping coefficient	Czeta(1) Czeta(2) Czeta(3))	moment/(rad/sec)
Lead-lag nonlinear damping coefficient	Vzeta(1) Vzeta(2) Vzeta(3)	moment/(rad/sec)^2
Fuselage linear damping coefficient in x and y direction	c(1) c(2)	force/(length/sec)
Fuselage nonlinear damping coefficient in x and y direction	v(1) v(2)	force/(length/sec)^2
Lead-lag linear spring coefficient	Ke(1) Ke(2) Ke(3)	moment/rad
Lead-lag nonlinear spring coefficient	Kd(1) Kd(2) Kd(3)	moment/rad^3
Lead-lag stop spring coefficient	Ks(1) Ks(2) Ks(3)	moment/rad
Effective fuselage stiffness in the x and y directions	K(1) K(2)	force/length
Fuselage states initial displacement conditions	xXi xYi	length
Fuselage states initial rate conditions	xrXi xrYi	length/sec
Blade states initial displacement conditions	x1i x2i x3i	rad
Blade states initial rate conditions	xr1i xr2i xr3i xr4i	rad/sec

Table 2. Parameter inputs for simple model

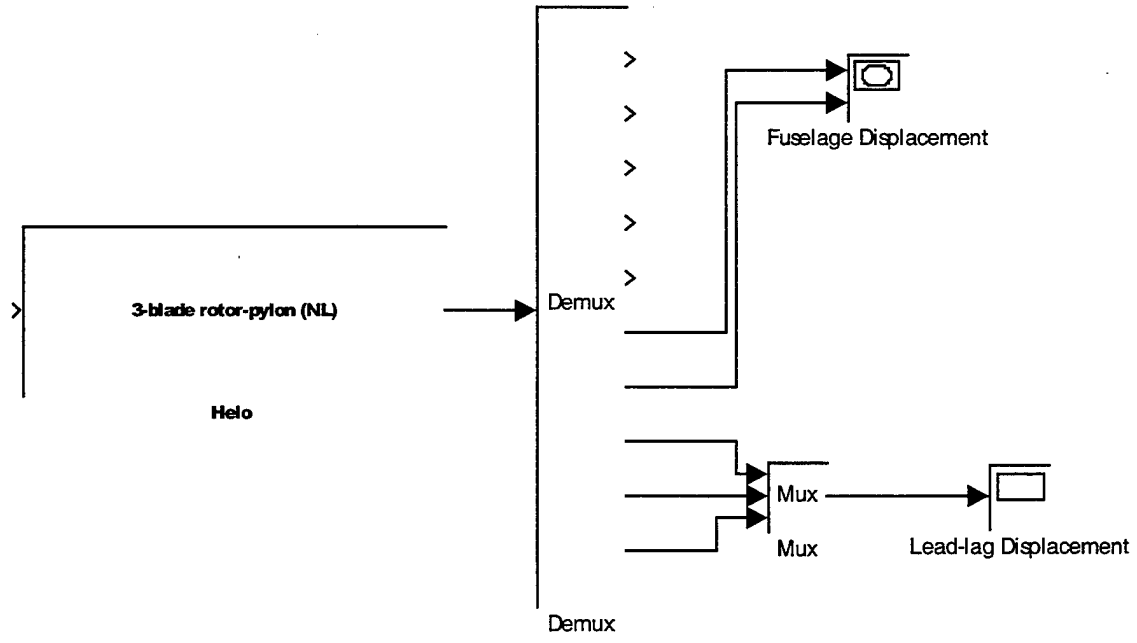

Figure 7. SIMULINK® diagram of NPS simple model

B. MODIFICATIONS TO SIMULINK® S-FUNCTION

Of significant importance in this and future versions of the simulation is MATLAB®'s definition of variables. From the original MATLAB® function where the B matrix was input as optimized C or Fortran code, the variables in the equations of motion were in array format. Using the Symbolic Math Toolbox requires the variables to be defined as symbolic. In doing so, an additional substitution step is required to solve the state derivatives as real variables. This substitution may be accomplished in either the derivation function or the MATLAB® S-function. A coordination between the two functions is required in order to properly derive the equations of motion and still run the simulation. For example time (t) is needed to derive the equations of motion, however it is not recognized in MATLAB® unless it is defined as a symbolic variable prior to the derivation, but once it is defined as a symbolic variable an actual value for 't' (each time step) may not be substituted in to the simulation model until it is redefined as a 'double array' variable.

Using a MATLAB® function to derive the equations of motion allows the model to call the derivation function with each time step. Unfortunately this process is very time consuming and severally slows the simulation process. The potential exists to cut and past the equations of motion into an S-function, in order to reduce the simulation run time. This method was not used in an attempt to maintain the integrity of the function calling process. Future thesis may have the opportunity to optimize this process. It should be noted that the cut and paste method would still significantly reduce the time, labor, and possible errors associated with converting the MAPLE® code.

As an alternative method to deriving the equation of motion with each time step, a data file was created with the state derivatives input from the derivation function. The goal of this setup was to reduce the computational time. This method was not found to be as effective as anticipated because the process still required the derived variables to be converted to 'double array' variables before value substitution in the S-function workspace.

C. USING THE SIMULINK MODEL

Six files are required to run the simulation. For the three bladed simple model the files are: simple3.m, helo3b5.m, simple.m, simple.mdl, signum.m, signum1.m, abs.m. From the MATLAB® command window type in

>> simple

a figure will appear identical to Figure 7. The operator may begin the simulation by "clicking" on the arrow (▶) on the menu bar. Two figures will be displayed, Fuselage Displacement and Lead-lag Displacement. The Fuselage Displacement figure represents the position of the fuselage in the x/y plane with respect time. The Lead-lag Displacement figure represents each blades position with respect to time. From here the operator can deduce whether the model is stable or unstable.

The rotor parameters may be changed by "double-clicking" on the box titled "3-blade rotor-pylon {NL}. A new window will appear containing the "Block Parameters." Each of these terms may be changed to effect the desired response from the model. An explanation of these terms is included in Table 2. Further explanation of using the NPS simulation model may be obtained in reference 17.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. RESULTS

A. EQUATIONS OF MOTION

The equations of motion calculated for the simple model by MATLAB®, Appendix A, are equivalent to the equations calculated by Robinson [Ref. 4]. As a result the plots produced by SIMULINK® display the same time history plots, Figures 8, 9. The derivation of the equations of motion using MATLAB® Symbolic Math Toolbox are complete.

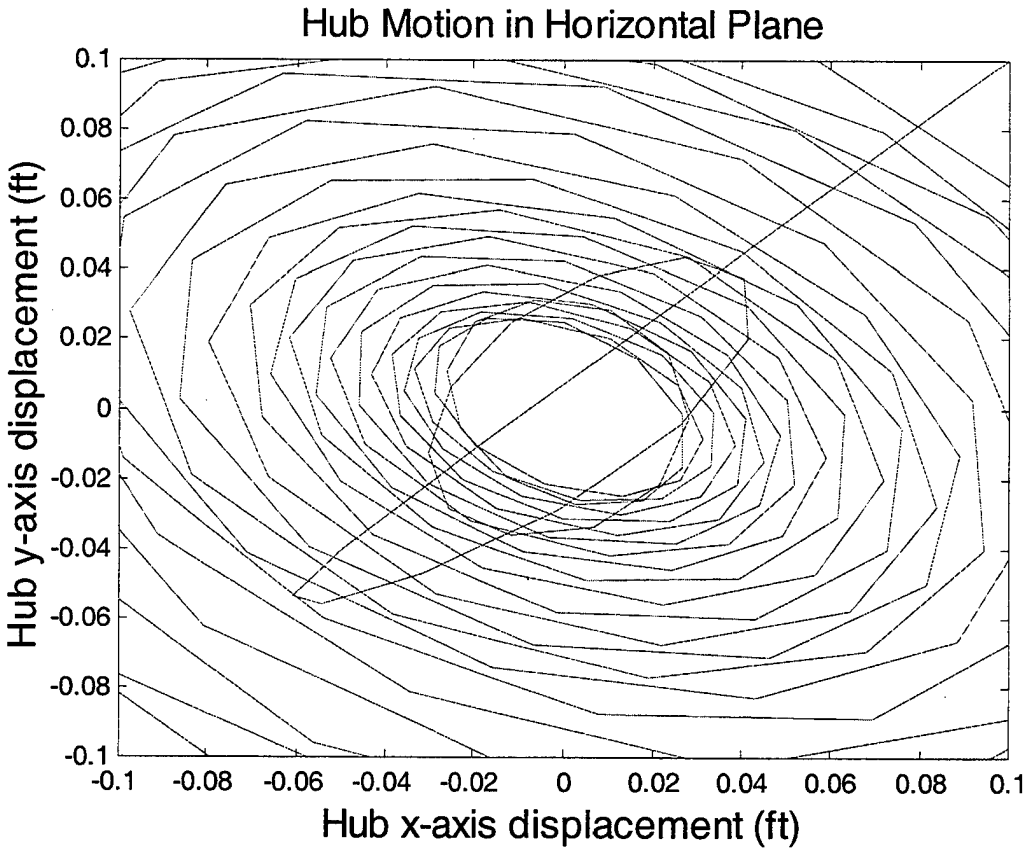

Figure 8. Fuselage displacement for unstable region

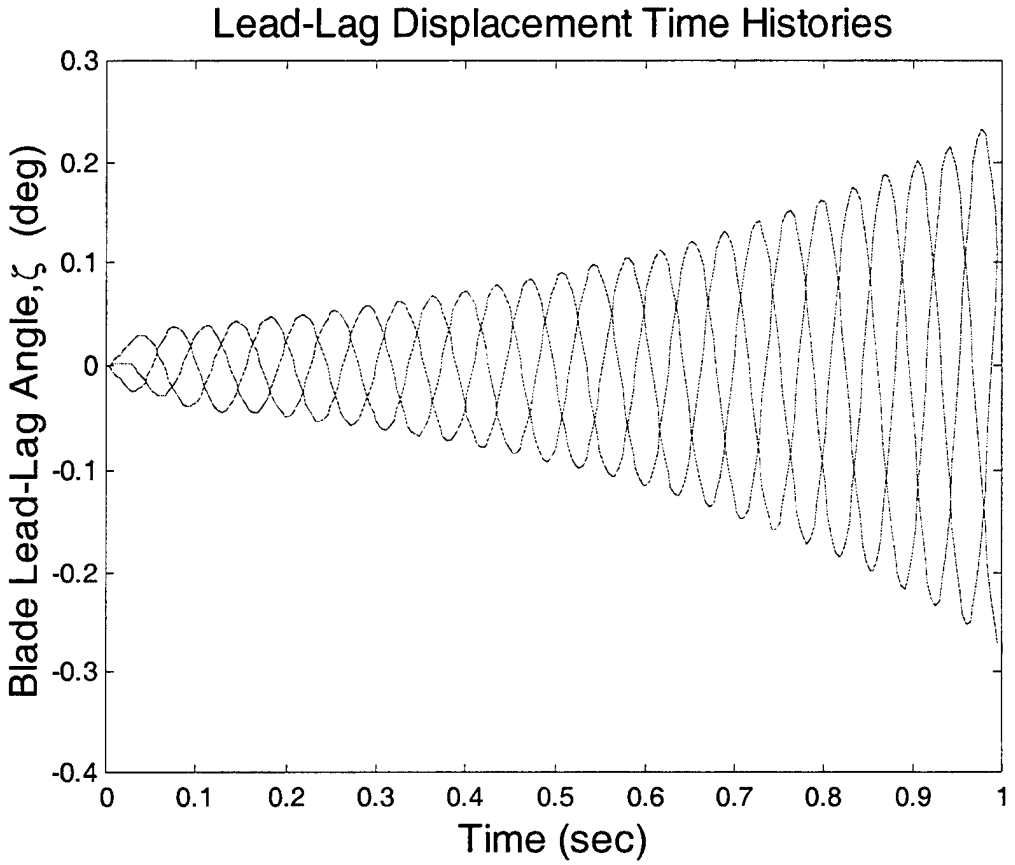

Figure 9. Lead-Lag displacement for unstable region

VII. CONCLUSIONS AND RECOMMENDATIONS

Previous work by Robinson and King was based on a derivation in which the equations of motion were derived in MAPLE® then converted into Fortran or C where the equations of motion were further converted in a MATLAB® usable format (Method A). The focus of this thesis was to derive the equations of motion using the Symbolic Math Toolbox in MATLAB®(Method B). This derivation allows direct communication between the derivation function and the simulation thus eliminating user interface in the simulation process.

Based on the simulation process utilized in this thesis, Method B, deriving the equations of motion using MATLAB® was found to possess both advantages and disadvantages.

ADVANTAGES

- No required conversion of equations of motion to MATLAB® readable code
- Eliminates "user" interface
- Single software package

DISADVANTAGES

- Time consuming
- Symbolic processing format is not as usable as MAPLE®
- Does not provide instant feedback to equation processing
- Requires additional substitution step

TRADEOFF

- It was found, using Method B, that eliminating user interface from the simulation process incurs a substantial run time penalty
- With minor user interface, modification to Method B eliminates the time penalty incurred
- The minor user interface still reduces the potential for error associated with code conversion
- Modification to Method B maintains a comparable run to Method A

Expanding on the "Tradeoff Option", if the user intends to utilize the NPS modeler in the form in which the simulation calls the derivation function internally, Method B, then a run time penalty is incurred. However if modifications to this process are acceptable, the result is a reduction in potential errors associated with code conversion and a savings in the time required to perform this process.

For example, potential modification to the simulation process used in Method B is the option of "cutting" the resultant matrices from the MATLAB® workspace and "pasting" them into the simulation workspace. This "cut and paste" method is similar to deriving the equations of motion in Method A. However, it eliminates the process of converting the code from Fortran or C to MATLAB® executable code thus reducing "user" interaction and eliminating the potential for user contamination of data.

Future emphasis should be placed on optimizing the passing of variables between the MATLAB® derivation function and the simulation process. Continued analysis of this process may determine a more effective means of substituting the required variables and reduce the time to run the simulation. With further optimization of the NPS modeler in the MATLAB® format and the continued increase in computational processing speeds, eliminating user interaction between the derivation and simulation process is a realistic goal.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Coleman, R. P., and Feingold, A. M., "Theory of Self Excited Mechanical Oscillations of Helicopter Rotors with Hinged Blades," NACA Report 1351, 1958.

2. Feingold, A. M., "Theory of Mechanical Oscillations of Rotors with Two Hinged Blades," NACA ARR no. 3113 Sept. 1943.

3. Deutsch, M. L., "Ground Vibrations of Helicopters," Journal of Aeronautical Sciences, Vol. 15, No. 5, May 1946, pp. 223-228.

4. Robinson, C. S., "Modeling and Analysis of Helicopter Ground Resonance Utilizing Symbolic Processing and Dynamic Simulation Software," Aeronautical Engineer's Thesis, Naval Postgraduate School, March 1997.

5. King, R. L., "Nonlinear Dynamic in the Modeling of Helicopter Rotor Blade Lead/Lag Motion" Aeronautical Engineering Dissertation, Naval Postgraduate School, June 1999.

6. Straub, F. K., "Study of Eliminate Ground Resonance Using Active Controls," NASA Contractor Report 166609, 1984.

7. Venkatesan, C. and Freidmann, P., "Aeroelastic Effects in Mulit-Rotor Vehicles With Application to Hybrid Heavy Lift System, Part I: Formulation of Equations of Motion," NASA Contractor Report 3822, 1984.

8. Tongue, B. H. and Flowers, G., "Nonlinear Rotorcraft Analysis Using Symbolic Manipulation," Applied Mathematical Modeling," Journal of Sound and Vibration, 1988, 122, pp. 233-241.

9. Tongue, B. H. and Flowers, G., "Nonlinear Rotorcraft Analysis," International Journal of Nonlinear Mechanics, Vol. 23, No 3, September 1987, pp. 189-203.

10. Rafanello, S. P., "Modeling the Coupled Rotor/Fuselage Response of the H-3 Sea King Utilitzing the NPS Full Nonlinear Response Simulation," Master of Science Thesis (Aeronautics), Naval Postgraduate School, March 1999.

11. Wood, E. R., "Preliminary Report on Mechanical Instability of HSS-2 Helicopters," Sikorsky Engineering Report No. 61163, 14 November 1958.

12. Robinson, C. S., Wood, E. R., and King, R. L., "Full Nonlinear Simulation of Coupled Rotor-Fuselage Response Using Symbolically Derived Equations of Motion," American Helicopter Society 54th Annual Forum and Technology Display, Washington DC, May 20-22, 1988.

13. Moler, C., Costa, P. J., "Matlab Symbolic Math Toolbox," User Guide, Ver. 2.0, the Mathworks, Inc., May 1997.

14. Warmbrodt, W., and Freidmann P., "Formulation of Coupled Rotor/Fuselage Equations of Motion,", Vertica, Vol. 3, pp. 245-271.

15. Deutsch, M. L., "Ground Vibrations of Helicopters," Journal of Aeronautical Sciences, Vol. 15, No 5, May 1946, pp. 223-228.

16. Wood, E. R., "Preliminary Report on Mechanical Instability of HSS-2 Helicopters," Sikorsky Aircraft Report No. SER-61163 November 1958.

17. King, R. L., "The Naval Postgraduate School Helicopter Rotor Dynamics Nonlinear Simulation User's Guide," 1999.

18. Robinson, C. S., Wood, E. R., King, R. L., "Analysis of Helicopter Rotor Instabilities Using Symbolic Processing and Control System Dynamics Software," European Rotorcraft 23rd Annual Forum, Dresden Germany, 16-18 September,1997.

19. Robinson, C. S., Wood, E. R., King, R. L., "Ground/Air Resonance Simulation of Helicopter Rotor Systems Based on Full Non-Linear Equations of Motion," Seventh International Workshop on Dynamics and Aeroelastic Stability Modeling of Rotorcraft Systems, Washington University and U.S. Army Research Office, St. Louis, MO, 14-16 October 1997.

20. Robinson, C. S., Wood, E. R., King, R. L., "Simulation of Helicopter Dynamic Mechanical Instability by MAPLE®-Based Nonlinear Lagrangian Derivation," AIAA/AHS/ASME/ASCE 39th Structures, Structural Dynamics, and Materials Conference, Long Beach CA, 20-23 April 1998.

21. King, R. L., Wood, E. R., "Results of Froude Scale RAH-66 Model Rotor Simulation Using the NPS Nonlinear Rotor Simulation," Eighth Annual Army Research Office Workshop on Aeroelasticity of Rotorcraft Systems, Penn State University, 18-20 October 1999.

22. King, R. L., Wood, E. R., "Interblade Coupling – An Alternate Approach to Improved Rotor Stability without Lag Dampers," Eighth Annual Army Research Office Workshop on Aeroelasticity of Rotorcraft Systems, Penn State University, 18-20 October 1999.

23. Tongue, B. H., "Limit Cycle Oscillations of a Nonlinear Rotorcraft Model," AIAA Journal, Vol. 22, July 1984, pp. 967-974.

24. Peters, D. A., Hohenemser, K. H., "Application of the Floquet Transition Matrix to Problems of Lifting Rotor Stability," *Journal of the American Helicopter Society*, April 1971, Vol. 16, pp. 25-33.

25. Hammond, C. E., "An Application of Floquet Theory to Prediction of Mechanical Instability," *Journal of the American Helicopter Society*, October 1974, pp.14-23.

26. Ormiston, R. A., "Rotor-Fuselage Dynamics of Helicopter Air and Ground Resonance", *Journal of the American Helicopter Society*, April 1991, pp. 3-20.

27. Venkatesan, C. and Friedmann, P., "Aeroelastic Effects in Multi-Rotor Vehicles With Application to Hybrid Heavy Lift System, Part I: Formulation of Equations of Motion," NASA Contractor Report 3822, 1984.

28. Tongue, B. H., and Jankowski, M. D., "Construction and Analysis of a Simplified Nonlinear Ground Resonance Model," *Journal of Sound and Vibration*, 1988, 122, pp. 233-241.

29. Tang, D. M., and Dowell, E. H., "Influence of Nonlinear Blade Damping on Helicopter Ground Resonance Instability," *Journal of Aircraft*, Vol. 23, No 2, February 1986, pp. 104-110.

30. Ormiston, R. A., "The Challenge of The Damperless Rotor.", Paper No. 68, 22nd European Rotorcraft Forum, Brighton, U.K. September 1996.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATLAB® EQUATIONS OF MOTION FOR SIMPLE INPLANE (COLEMAN) MODEL WITH THREE ROTOR BLADES

pretty(EOM)

```
[       2
[1/2 v1 dq1  abs(1, dq1) + v1 dq1 | dq1 | + M1 ddq1

                                    2
 - 2 mb2 R %4 Omega cos(q4) dq4 - mb2 R %4 Omega  cos(q4)

           2                        2
 - mb1 R %6 Omega  cos(q3) - mb1 R %6 cos(q3) dq3

                              2
 - mb1 R %6 sin(q3) ddq3 + mb2 R %3 Omega  sin(q4)

                 2
 + mb2 R %3 sin(q4) dq4  - mb2 R %3 cos(q4) ddq4

           2                        2
 - mb3 R %2 Omega  cos(q5) - mb3 R %2 cos(q5) dq5

                              2
 - mb3 R %2 sin(q5) ddq5 + mb1 R %5 Omega  sin(q3)

                 2
 + mb1 R %5 sin(q3) dq3  - mb3 R %1 cos(q5) ddq5

                              2
 - mb1 R %5 cos(q3) ddq3 + mb3 R %1 Omega  sin(q5)

                 2                  2
 + mb3 R %1 sin(q5) dq5  - mb2 R %4 cos(q4) dq4

 - mb2 R %4 sin(q4) ddq4 + K1 q1 + mb1 ddq1 + mb2 ddq1 + c1 dq1

 + mb3 ddq1 + 2 mb1 R %5 Omega sin(q3) dq3

 - 2 mb1 R %6 Omega cos(q3) dq3 + 2 mb3 R %1 Omega sin(q5) dq5

 - 2 mb3 R %2 Omega cos(q5) dq5 + 2 mb2 R %3 Omega sin(q4) dq4
```

```
                2              2              2
- mb3 e1 %2 Omega  - mb2 e1 %4 Omega  - mb1 e1 %6 Omega  ,

      2                                  2
1/2 v2 dq2  abs(1, dq2) + v2 dq2 | dq2 | - mb3 R %1 Omega  cos(q5)

                2
- mb1 R %6 sin(q3) dq3  + mb1 R %6 cos(q3) ddq3

              2
- mb1 R %5 Omega  cos(q3) - mb1 R %5 sin(q3) ddq3

              2                          2
- mb2 R %3 Omega  cos(q4) - mb1 R %5 cos(q3) dq3

+ mb2 R %4 cos(q4) ddq4 + mb3 R %2 cos(q5) ddq5

                2             2
- mb3 R %2 sin(q5) dq5  - mb3 R %2 Omega  sin(q5)

                2
- mb3 R %1 cos(q5) dq5  - mb2 R %3 sin(q4) ddq4

              2                                2
- mb1 R %6 Omega  sin(q3) + mb1 ddq2 - mb2 R %3 cos(q4) dq4

                                   2
- mb3 R %1 sin(q5) ddq5 - mb2 R %4 sin(q4) dq4

              2                    2              2
- mb2 R %4 Omega  sin(q4) - mb1 e1 %5 Omega  - mb3 e1 %1 Omega

+ mb3 ddq2 + mb2 ddq2 - 2 mb3 R %1 Omega cos(q5) dq5

- 2 mb1 R %5 Omega cos(q3) dq3 - 2 mb1 R %6 Omega sin(q3) dq3

- 2 mb3 R %2 Omega sin(q5) dq5 - 2 mb2 R %4 Omega sin(q4) dq4

- 2 mb2 R %3 Omega cos(q4) dq4 + M2 ddq2 + K2 q2 + c2 dq2

                2             2
- mb2 e1 %3 Omega  , mb1 e1 Omega  R sin(q3) + Czeta1 dq3 + Ke1 q3
```

```
      3          2                    2
+ Kd1 q3  + Vzeta1 dq3  abs(1, dq3) - u1(t) + mb1 R  ddq3

+ mb1 ddq2 R %6 cos(q3) - mb1 ddq2 R %5 sin(q3)

- mb1 ddq1 R %6 sin(q3) - mb1 ddq1 R %5 cos(q3)

                          2
+ 2 Vzeta1 dq3 | dq3 | , Vzeta2 dq4  abs(1, dq4)

                        3
+ 2 Vzeta2 dq4 | dq4 | + Kd2 q4  + Ke2 q4 - u2(t)

          2
+ mb2 e1 Omega  R sin(q4) - mb2 ddq1 R %4 sin(q4)

                       2
- mb2 ddq1 R %3 cos(q4) + mb2 R  ddq4 + mb2 ddq2 R %4 cos(q4)

                                          3
- mb2 ddq2 R %3 sin(q4) + Czeta2 dq4 , Czeta3 dq5 + Ke3 q5 + Kd3 q5

             2
- u3 + Vzeta3 dq5  abs(1, dq5) + 2 Vzeta3 dq5 | dq5 |

                                      2
- mb3 ddq1 R %2 sin(q5) - mb3 ddq1 R %1 cos(q5) + mb3 R  ddq5

+ mb3 ddq2 R %2 cos(q5) - mb3 ddq2 R %1 sin(q5)

           2        ]
+ mb3 e1 Omega  R sin(q5)]

%1 := sin(Omega t + Phi3)

%2 := cos(Omega t + Phi3)

%3 := sin(Omega t + Phi2)

%4 := cos(Omega t + Phi2)

%5 := sin(Omega t + Phi1)

%6 := cos(Omega t + Phi1)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB® WORKSHEET FOR SIMPLE INPLANE (COLEMAN) MODEL WITH THREE ROTOR BLADES

```
function [A1, f1] = simple3()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EQUATIONS OF MOTION FOR A HELICOPTER IN GROUND RESONANCE
% CONSIDERING ONLY INPLANE DEGREES OF FREEDOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definition of variables for transformations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms Phi1 Phi2 Phi3 Omega t

Phi = [Phi1 Phi2 Phi3];
psi = Omega*t + Phi;

zet1 = sym('zet1(t)');
zet2 = sym('zet2(t)');
zet3 = sym('zet3(t)');
zet = [zet1 zet2 zet3];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COORDINATE TRANSFORMATIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:3
   m1(:,:,i) = [cos(psi(i)) sin(psi(i)) 0
      -sin(psi(i)) cos(psi(i)) 0
      0 0 1];
   m2(:,:,i) = [cos(zet(i)) sin(zet(i)) 0
      -sin(zet(i)) cos(zet(i)) 0
      0 0 1];
end
m1(:) = m1(:).';
m2(:) = m2(:).';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    ENERGY OF ROTOR BLADES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kinetic energy of rotor blade (TBk)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms e1 R mb1 mb2 mb3

mb = [mb1 mb2 mb3];
u1 = sym('u1(t)');
u2 = sym('u2(t)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rhoHI_I = [u1 u2 0];
rhoBuH = [e1 0 0];
rhoPBd = [R 0 0];

for i = 1:3
   rhoBuH_I = rhoBuH*m1(:,:,i);
   rhoPBd_I = rhoPBd*m1(:,:,i)*m2(:,:,i);
   rho = simplify(rhoBuH_I + rhoPBd_I + rhoHI_I).';
   V(:,:,i) = diff(rho,t).';
   TBk(:,:,i) = 1/2*mb(i)*(V(1,:,i)^2 + V(2,:,i)^2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Potential energy or rotor blade (UBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms Ke1 Ke2 Ke3 Kd1 Kd2 Kd3 Ks1 Ks2 Ks3 z
Ke = [Ke1 Ke2 Ke3];
Kd = [Kd1 Kd2 Kd3];
Ks = [Ks1 Ks2 Ks3];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:3
   UBk1 = 1/2*Ke(i)*zet(i)^2;
   UBk2 = 1/4*Kd(i)*zet(i)^4;
   UBk3 = 1/4*Ks(i)*signum(zet(i)-z)*(zet(i)^2+z^2-
2*zet(i)*z)+1/4*Ks(i)*signum(zet(i)+z)*(-zet(i)^2-z^2-
2*zet(i)*z)+1/2*Ks(i)*zet(i)^2 + 1/2*Ks(i)*z^2;
   UBk(i) = UBk1 + UBk2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dissapative function of rotor blades (DBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms DB1 DB2 DB3 Czeta1 Czeta2 Czeta3 Vzeta1 Vzeta2 Vzeta3
Czeta = [Czeta1 Czeta2 Czeta3];
Vzeta = [Vzeta1 Vzeta2 Vzeta3];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for i = 1:3
   DBk(i) = 1/2*Czeta(i)*(diff(zet(i),t))^2 +
Vzeta(i)*(diff(zet(i),t))^2*abs(diff(zet(i),t));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          ENERGY OF HUB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kinetic energy TH) / Potential energy (UH) / Dissapative
function (DH)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms K1 K2 c1 c2 v1 v2 M1 M2
c = [c1 c2];
v = [v1 v2];
M = [M1 M2];
K = [K1 K2];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

TF = 1/2*M(1)*diff(u1,t)^2 + 1/2*M(2)*diff(u2,t)^2;
UF = 1/2*K(1)*u1^2 + 1/2*K(2)*u2^2;
DF = 1/2*c(1)*(diff(u1,t))^2 +
1/2*c(2)*(diff(u2,t))^2+1/2*v(1)*(diff(u1,t))^2*abs(diff(u1
,t))+1/2*v(2)*(diff(u2,t))^2*abs(diff(u2,t));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generalized forces on generalized displacements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms F1 F2 F3 F4 F5 u3
u = [u1 u2 u3];
F1 = 0; F2 = 0; F3 = u1; F4 = u2; F5 = u3;
F = [F1 F2 F3 F4 F5];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DOFF = [u1,u2];
DOFB = [zet(1), zet(2), zet(3)]; % or zet(1) and zet(2)
DOF = [DOFF, DOFB];
dDOF = diff(DOF,t);
ddDOF = diff(dDOF,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DERIVATION OF EQUATIONS OF MOTION USING LAGRANGE'S
EQUATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
syms q1 q2 q3 q4 q5 dq1 dq2 dq3 dq4 dq5 ddq1 ddq2 ddq3 ddq4
ddq5
DOFq = [ q1 q2 q3 q4 q5];
dDOFq = [ dq1 dq2 dq3 dq4 dq5];
ddDOFq = [ ddq1 ddq2 ddq3 ddq4 ddq5];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Substition set construction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set1 = [];
set2 = [];
for i=1:5 % size of DOFq vector
   set1 = [set1 maple(''='',DOF(i),DOFq(i)) ];
   set1 = [set1 maple(''='',dDOF(i),dDOFq(i)) ];
   set1 = [set1 maple(''='',ddDOF(i),ddDOFq(i)) ];
   set2 = [set2 maple(''='',DOFq(i),DOF(i)) ];
   set2 = [set2 maple(''='',dDOFq(i),dDOF(i)) ];
   set2 = [set2 maple(''='',ddDOFq(i),ddDOF(i)) ];
end
set1 = maple('convert',set1,'set');
set2 = maple('convert',set2,'set');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T = TF;
U = UF;
D1 = DF;

[l,n] = size(DOFB);
for i = 1:3
   T = T+TBk(:,:,i);
   U = U+UBk(i);
   D1 = D1+DBk(i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Switch of time dependent terms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Temp = maple('subs',set1,T);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Complete EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[l,n] = size(DOF);
for i = 1:5
```

```
   temp1 = diff(Temp,dDOFq(i));
   temp2 = maple('subs',set2,temp1);
   temp3 = diff(temp2,t);
   L1 = maple('subs',set1,temp3);
   L2 = diff(Temp,DOFq(i));
   tempL3 = maple('subs',set1,U);
   L3 = diff(tempL3,DOFq(i)); % check dimensions of DOFq
   tempL4 = maple('subs',set1,D1);
   L4 = diff(tempL4,dDOFq(i)); % check dimension of DOFq
   EOM(i) = simplify(L1-L2+L3+L4-F(i));  % check dimension
of EOM and F
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Elimination of excess terms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tempS = [ signum1(1,q3-z) signum1(1,q4-z) signum1(1,q5-z)
signum1(1,q3+z) signum1(1,q4+z) signum1(1,q5+z)
maple('abs',1,dq3) maple('abs',1,dq4) maple('abs',1,dq5)
maple('abs',1,dq1) maple('abs',1,dq2) ];
temp0 = [ 0 0 0 0 0 0 0 0 0 0 0 ];
setS = [];

for i=1:11
   setS = [setS maple('`=`',tempS(i),temp0(i)) ];
end
setS = maple('convert',setS,'set');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              GENERATE A1 AND f1 FROM EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine second derivative terms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:n
   for j = 1:n
      A(i,j) = maple('coeff',EOM(i),ddDOFq(j));
      A(i,j) = maple('subs',setS,A(i,j));
   end
end

Ax2dot = ddDOFq*A;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eliminate second derivative terms from EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for i = 1:n
   f(i) = (EOM(i)-Ax2dot(i));
   f(i) = maple('subs',setS,f(i));
   f(i) = -(simplify(f(i)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate state vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X1dot = [x1 x2 x3 x4 x5];
X1temp = [x6 x7 x8 x9 x10];
X1 = [X1temp X1dot];
DOFqtemp = [DOFq dDOFq];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert terms to state vector form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

setX = [];
for i=1:10
   setX = [setX maple('`=`',DOFqtemp(i),X1(i)) ];
end
setX = maple('convert',setX,'set');

A1 = maple('subs',setX,A);
f1 = maple('subs',setX,f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eliminate variable 't' from input 'u'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms u1 u2 u3
ut = [u1 u2 u3];
A1 = subs(A1,u,ut);
f1 = subs(f1,u,ut);
```

APPENDIX C. SIMULINK® S-FUNCTION FOR SIMPLE INPLANE (COLEMAN) MODEL WITH THREE ROTOR BLADES

```
function [sys, x0] = helo3b5(t,x,u,flag,I1,I2,I3,I4,I5,I6);

% function [sys, x0] =
helo3bA(t,x,u,flag,I1,I2,I3,I4,I5,I6)
%
%   S-function arguments:
%   ---------------------
%   t      =  time
%   x      =  state vector
%   u      =  input vector
%   flag   =  switch used by numerical integration
(simulation)
%             routine to access certain parts of the s-
function
%
%   S-function input parameters:
%   ----------------------------
%
%   I1     =  [mb(1),mb(2),mb(3),M(1),M(2)]
%
%   I2     =  [R,Omega,e1,z]
%
%   I3     =  [Phi(1),Phi(2),Phi(3)]
%
%   I4     =  [c(1),c(2),v(1),v(2),
%              Czeta(1),Czeta(2),Czeta(3),
%              Vzeta(1),Vzeta(2),Vzeta(3)]
%
%   I5     =  [Ke(1),Ke(2),Ke(3),
%              Kd(1),Kd(2),Kd(3),
%              Ks(1),Ks(2),Ks(3),
%              K(1),K(2)]
%
%   I6     =  [xrXi,xrYi,xr1i,xr2i,xr3i,
%              xXi,xYi,x1i,x2i,x3i]
%
% S-function to represent dynamics of 3 bladed coupled
rotor-
% fuselage model which considers only inplane degrees of
```

```
% freedom, i.e., x and y translational fuselage degrees of
freedom
% and lead-lag rotor blade degrees of freedom.
%
% Explaination of variables:
% -------------------------
%
% mb      ->    mass of blade
% M       ->    effective mass of fuselage
% R       ->    distance from lead-lag hinge to blade center
of mass
% e1      ->    blade hinge offset
% Omega   ->    rotor speed
% z       ->    angle at which blade hits stops
% Phi     ->    blade phase angle w.r.t. azimuth postion
% c       ->    fuselage linear damping
% v       ->    fuselage hydraulic damping
% Czeta   ->    blade linear damping
% Vzeta   ->    blade hydraulic damping
% K       ->    effective stiffness of fuselage (landing
gear stiffness)
% Ke      ->    blade elastic spring constant
% Kd      ->    blade duffing spring constant
% Ks      ->    blade stop effective spring constant
% xr_ _i  ->    initial rate
% x_ _i   ->    initial displacement
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define input parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%mb1=I1(1);mb2=I1(2);mb3=I1(3);M1=I1(4);M2=I1(5);
%R=I2(1);Omega=I2(2);e1=I2(3);z=I2(4);
%Phi1=I3(1);Phi2=I3(2);Phi3=I3(3);
%c1=I4(1);c2=I4(2);v1=I4(3);v2=I4(4);
%Czeta1=I4(5);Czeta2=I4(6);Czeta3=I4(7);
%Vzeta1=I4(8);Vzeta2=I4(9);Vzeta3=I4(10);
%Ke1=I5(1);Ke2=I5(2);Ke3=I5(3);
%Kd1=I5(4);Kd2=I5(5);Kd3=I5(6);
%Ks1=I5(7);Ks2=I5(8);Ks3=I5(9);
%K1=I5(10);K2=I5(11);
xrXi=I6(1);xrYi=I6(2);xr1i=I6(3);xr2i=I6(4);xr3i=I6(5);
xXi=I6(6);xYi=I6(7);x1i=I6(8);x2i=I6(9);x3i=I6(10);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% S-function flag conditionals
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if flag == 0

      sys=[10,0,10,3,0,0];
      x0=[xrXi,xrYi,xr1i,xr2i,xr3i,xXi,xYi,x1i,x2i,x3i];

elseif abs(flag) == 1

% Calculate derivatives

[A1, f1] = simple3;

x1 = x(1); x2 = x(2); x3 = x(3); x4 = x(4); x5 = x(5); x6 =
x(6); x7 = x(7); x8 = x(8); x9 = x(9); x10 = x(10);
u1 = u(1); u2 = u(2); u3 = u(3);
x = subs(x);
u = subs(u);
f1 = subs(f1);
A1 = subs(A1);

sys = zeros(1,2*5);
sys(1:5) = A1\f1.';
sys(6:10) = x(1:5);


% Output states

elseif abs(flag) == 3

 %     psi=Omega*t;
 %     psi1=psi+Phi(1);
 %     psi2=psi+Phi(2);
 %     psi3=psi+Phi(3);
 %     xc1=e1*cos(psi1)+R*cos(psi1+x(8));
 %     xc2=e1*cos(psi2)+R*cos(psi2+x(9));
 %     xc3=e1*cos(psi3)+R*cos(psi3+x(10));
 %     yc1=e1*sin(psi1)+R*sin(psi1+x(8));
 %     yc2=e1*sin(psi2)+R*sin(psi2+x(9));
 %     yc3=e1*sin(psi3)+R*sin(psi3+x(10));
 %     xc=(mb(1)*xc1+mb(2)*xc2+mb(3)*xc3)/(sum(mb));
 %     yc=(mb(1)*yc1+mb(2)*yc2+mb(3)*yc3)/(sum(mb));
 %     Mag=sqrt(xc^2+yc^2);
 %     if (xc < 0 & yc < 0) | (yc < 0)
 %        theta=pi+atan(yc/xc);
```

```
%      else
%         theta=atan(yc/xc);
%      end
%      gamma=theta+pi;
%      alpha1=(psi1/(2*pi)-floor(psi1/(2*pi)))*2*pi;
%      alpha2=(psi2/(2*pi)-floor(psi2/(2*pi)))*2*pi;
%      alpha3=(psi3/(2*pi)-floor(psi3/(2*pi)))*2*pi;
      sys(1:10)=x;
%      sys(11)=R*Mag*sin(gamma-alpha1);
%      sys(12)=R*Mag*sin(gamma-alpha2);
%      sys(13)=R*Mag*sin(gamma-alpha3);
%      sys(14)=theta;
else

      sys = [];

end
```

APPENDIX D. MATLAB® WORKSHEET FOR SIMPLE INPLANE MOTION WITH VARIABLE NUMBERS OF BLADES

```
function [A1, f1] = simple()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EQUATIONS OF MOTION FOR A HELICOPTER IN GROUND RESONANCE
% CONSIDERING ONLY INPLANE DEGREES OF FREEDOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% mb      ->    mass of blade
% M       ->    effective mass of fuselage
% R       ->    distance from lead-lag hinge to blade center
of mass
% e1      ->    blade hinge offset
% Omega   ->    rotor speed
% z       ->    angle at which blade hits stops
% Phi     ->    blade phase angle w.r.t. azimuth postion
% c       ->    fuselage linear damping
% v       ->    fuselage hydraulic damping
% Czeta   ->    blade linear damping
% Vzeta   ->    blade hydraulic damping
% K       ->    effective stiffness of fuselage (landing
gear stiffness)
% Ke      ->    blade elastic spring constant
% Kd      ->    blade duffing spring constant
% Ks      ->    blade stop effective spring constant
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear
clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define transformation variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k = 3;                                  % number of blades

Phit = [Phi1 Phi2 Phi3 Phi4 Phi5 Phi6 Phi7];
Phi = Phit(1:k);
psi = Omega*t + Phi;                    % blade position angle

zet1 = sym('zet1(t)'); zet2 = sym('zet2(t)'); zet3 =
sym('zet3(t)');
```

```
zet4 = sym('zet4(t)'); zet5 = sym('zet5(t)'); zet6 =
sym('zet6(t)');
zet7 = sym('zet7(t)');
zett = [zet1 zet2 zet3 zet4 zet5 zet6 zet7];
zet = zett(1:k);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COORDINATE TRANSFORMATIONS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:k
   m1(:,:,i) = [cos(psi(i)) sin(psi(i)) 0
      -sin(psi(i)) cos(psi(i)) 0
      0 0 1];                          % first transformation
   m2(:,:,i) = [cos(zet(i)) sin(zet(i)) 0
      -sin(zet(i)) cos(zet(i)) 0
      0 0 1];                         % second transformation
end
m1(:) = m1(:).';
m2(:) = m2(:).';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       ENERGY OF ROTOR BLADES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kinetic energy of rotor blade (TBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms e1 R mb1 mb2 mb3 mb4 mb5 mb6 mb7
mbt = [mb1 mb2 mb3 mb4 mb5 mb6 mb7];
mb = mbt(1:k);
u1 = sym('u1(t)'); u2 = sym('u2(t)'); u3 = sym('u3(t)'); u4
= sym('u4(t)');
u5 = sym('u5(t)'); u6 = sym('u6(t)'); u7 = sym('u7(t)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

rhoHI_I = [u1 u2 0];
rhoBuH = [e1 0 0];
rhoPBd = [R 0 0];

for i = 1:k
   rhoBuH_I = rhoBuH*m1(:,:,i);
   rhoPBd_I = rhoPBd*m1(:,:,i)*m2(:,:,i);
   rho = simplify(rhoBuH_I + rhoPBd_I + rhoHI_I).';
   V(:,:,i) = diff(rho,t).';
   TBk(:,:,i) = 1/2*mb(i)*(V(1,:,i)^2 + V(2,:,i)^2);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Potential energy or rotor blade (UBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms Ke1 Ke2 Ke3 Ke4 Ke5 Ke6 Ke7
syms Kd1 Kd2 Kd3 Kd4 Kd5 Kd6 Kd7
syms Ks1 Ks2 Ks3 Ks4 Ks5 Ks6 Ks7 z
Ket = [Ke1 Ke2 Ke3 Ke4 Ke5 Ke6 Ke7];
Ke = Ket(1:k);
Kdt = [Kd1 Kd2 Kd3 Kd4 Kd5 Kd6 Kd7];
Kd = Kdt(1:k);
Kst = [Ks1 Ks2 Ks3 Ks4 Ks5 Ks6 Ks7];
Ks = Kst(1:k);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:k
   UBk1 = 1/2*Ke(i)*zet(i)^2;
   UBk2 = 1/4*Kd(i)*zet(i)^4;
   %UBk3 = 1/4*Ks(i)*signum(zet(i)-z)*(zet(i)^2+z^2-
2*zet(i)*z)+1/4*Ks(i)*signum(zet(i)+z)*(-zet(i)^2-z^2-
2*zet(i)*z)+1/2*Ks(i)*zet(i)^2 + 1/2*Ks(i)*z^2;
   UBk(i) = UBk1 + UBk2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Dissapative function of rotor blades (DBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms DB1 DB2 DB3 DB4 DB5 DB6 DB7
syms Czeta1 Czeta2 Czeta3 Czeta4 Czeta5 Czeta6 Czeta7
syms Vzeta1 Vzeta2 Vzeta3 Vzeta4 Vzeta5 Vzeta6 Vzeta7
Czetat = [Czeta1 Czeta2 Czeta3 Czeta4 Czeta5 Czeta6
Czeta7];
Czeta = Czetat(1:k);
Vzetat = [Vzeta1 Vzeta2 Vzeta3 Vzeta4 Vzeta5 Vzeta6
Vzeta7];
Vzeta = Vzetat(1:k);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 1:3
   DBk(i) = 1/2*Czeta(i)*(diff(zet(i),t))^2 +
Vzeta(i)*(diff(zet(i),t))^2*abs(diff(zet(i),t));
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        ENERGY OF HUB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kinetic energy TH) / Potential energy (UH) / Dissapative
function (DH)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms K1 K2 c1 c2 v1 v2 M1 M2
c = [c1 c2];
v = [v1 v2];
M = [M1 M2];
K = [K1 K2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TF = 1/2*M(1)*diff(u1,t)^2 + 1/2*M(2)*diff(u2,t)^2;
UF = 1/2*K(1)*u1^2 + 1/2*K(2)*u2^2;
DF = 1/2*c(1)*(diff(u1,t))^2 +
1/2*c(2)*(diff(u2,t))^2+1/2*v(1)*(diff(u1,t))^2*abs(diff(u1
,t))+1/2*v(2)*(diff(u2,t))^2*abs(diff(u2,t));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generalized forces on generalized displacements
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms F1 F2 F3 F4 F5 u3
ut = [u1 u2 u3 u4 u5 u6 u7];
u = ut(1:k);
F1 = 0; F2 = 0; F3 = u1; F4 = u2; F5 = u3; F6 = u4; F7 =
u5; F8 = u6; F9 = u7;
F = [F1 F2 u];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DOFF = [u1,u2];
DOFB = zet; % or zet(1) and zet(2)
DOF = [DOFF, DOFB];
dDOF = diff(DOF,t);
ddDOF = diff(dDOF,t);
[l,n] = size(DOF);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DERIVATION OF EQUATIONS OF MOTION USING LAGRANGE'S
EQUATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms q1 q2 q3 q4 q5 q6 q7 q8 q9
```

```
syms dq1 dq2 dq3 dq4 dq5 dq6 dq7 dq8 dq9
syms ddq1 ddq2 ddq3 ddq4 ddq5 ddq6 ddq7 ddq8 ddq9
DOFqt = [ q1 q2 q3 q4 q5 q6 q7 q8 q9];
DOFq = DOFqt(1:n);
dDOFqt = [ dq1 dq2 dq3 dq4 dq5 dq6 dq7 dq8 dq9];
dDOFq = dDOFqt(1:n);
ddDOFqt = [ ddq1 ddq2 ddq3 ddq4 ddq5 ddq6 ddq7 ddq8 ddq9];
ddDOFq = ddDOFqt(1:n);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Substution set construction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set1 = [];
set2 = [];
for i=1:n % size of DOFq vector
   set1 = [set1 maple('`=`',DOF(i),DOFq(i)) ];
   set1 = [set1 maple('`=`',dDOF(i),dDOFq(i)) ];
   set1 = [set1 maple('`=`',ddDOF(i),ddDOFq(i)) ];
   set2 = [set2 maple('`=`',DOFq(i),DOF(i)) ];
   set2 = [set2 maple('`=`',dDOFq(i),dDOF(i)) ];
   set2 = [set2 maple('`=`',ddDOFq(i),ddDOF(i)) ];
end
set1 = maple('convert',set1,'set');
set2 = maple('convert',set2,'set');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T = TF;
U = UF;
D1 = DF;

for i = 1:k
   T = T+TBk(:,:,i);
   U = U+UBk(i);
   D1 = D1+DBk(i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Switch of time dependent terms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Temp = maple('subs',set1,T);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Complete EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[1,n] = size(DOF);
```

```
for i = 1:n
   temp1 = diff(Temp,dDOFq(i));
   temp2 = maple('subs',set2,temp1);
   temp3 = diff(temp2,t);
   L1 = maple('subs',set1,temp3);
   L2 = diff(Temp,DOFq(i));
   tempL3 = maple('subs',set1,U);
   L3 = diff(tempL3,DOFq(i)); % check dimensions of DOFq
   tempL4 = maple('subs',set1,D1);
   L4 = diff(tempL4,dDOFq(i)); % check dimension of DOFq
   EOM(i) = simplify(L1-L2+L3+L4-F(i));  % check dimension
of EOM and F
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Elimination of excess baggage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tempSmin = [ signum1(1,q3-z) signum1(1,q4-z) signum1(1,q5-
z) signum1(1,q6-z) signum1(1,q7-z) signum1(1,q8-z)
signum1(1,q9-z) ];
tempSpls = [ signum1(1,q3+z) signum1(1,q4+z)
signum1(1,q5+z) signum1(1,q6+z) signum1(1,q7+z)
signum1(1,q8+z) signum1(1,q9+z) ];
tempSabs = [ maple('abs',1,dq1) maple('abs',1,dq2)
maple('abs',1,dq3) maple('abs',1,dq4) maple('abs',1,dq5)
maple('abs',1,dq6) maple('abs',1,dq7) maple('abs',1,dq8)
maple('abs',1,dq9)];
tempS = [tempSmin(1:n) tempSpls(1:n) tempSabs(1:n) ];
[g,p] = size(tempS);
temp0 = zeros(1,p);
setS = [];

for i=1:p
   setS = [setS maple('`=`',tempS(i),temp0(i)) ];
end
setS = maple('convert',setS,'set');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  GENERATE A1 AND f1 FROM EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Determine second derivative terms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:n
   for j = 1:n
      A(i,j) = maple('coeff',EOM(i),ddDOFq(j));
```

```
      A(i,j) = maple('subs',setS,A(i,j));
   end
end

Ax2dot = ddDOFq*A;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eliminate second derivative terms from EOM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1:n
   f(i) = (EOM(i)-Ax2dot(i));
   f(i) = maple('subs',setS,f(i));
   f(i) = -(simplify(f(i)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate state vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
X1dot = [x1 x2 x3 x4 x5];
X1temp = [x6 x7 x8 x9 x10];
X1 = [X1temp X1dot];
DOFqtemp = [DOFq dDOFq];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Convert terms to state vector form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setX = [];
for i=1:2*n
   setX = [setX maple('`=`',DOFqtemp(i),X1(i)) ];
end
setX = maple('convert',setX,'set');

A1 = maple('subs',setX,A);
f1 = maple('subs',setX,f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eliminate variable 't' from input 'u'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
syms u1 u2 u3
ut = [u1 u2 u3];
A1 = subs(A1,u,ut);
f1 = subs(f1,u,ut);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. MATLAB® WORKSHEET FOR COMPLEX (STRAUB) MODEL WITH VARIABLE NUMBER OF ROTOR BLADES

```
function [A1, f1] = compmod()

clear
clc

k = 3;  % number of blades, up to 7
syms Phi1 Phi2 Phi3 Phi4 Phi5 Phi6 Phi7 Omega t alpha

tempPhi = [Phi1 Phi2 Phi3 Phi4 Phi5 Phi6 Phi7];
Phi = tempPhi(1:k);
psi = Omega*t + Phi;

r1 = sym('r1(t)'); r2 = sym('r2(t)');
r = [r1 r2];

T1r = [1 0 0; 0 cos(r1) sin(r1); 0 -sin(r1) cos(r1)];
T2r = [cos(r2) 0 sin(r2); 0 1 0; -sin(r2) 0 cos(r2)];
m1 =  (T1r*T2r).'

for i = 1:k
   m2(:,:,i) = [cos(psi(i)) sin(psi(i)) 0; -sin(psi(i))
cos(psi(i)) 0; 0 0 1];
end

zet1 = sym('zet1(t)'); zet2 = sym('zet2(t)'); zet3 =
sym('zet3(t)'); zet4 = sym('zet4(t)');
zet5 = sym('zet6(t)'); zet6 = sym('zet6(t)'); zet7 =
sym('zet7(t)');
tempzet = [ zet1 zet2 zet3 zet4 zet5 zet6 zet7 ];
zet = tempzet(1:k);
bet1 = sym('bet1(t)'); bet2 = sym('bet2(t)'); bet3 =
sym('bet3(t)'); bet4 = sym('bet4(t)');
bet5 = sym('bet6(t)'); bet6 = sym('bet6(t)'); bet7 =
sym('bet7(t)');
tempbet = [ bet1 bet2 bet3 bet4 bet5 bet6 bet7 ];
bet = tempbet(1:k);


for i = 1:k
   T3z(:,:,i) = [cos(zet(i)) sin(zet(i)) 0; -sin(zet(i))
cos(zet(i)) 0; 0 0 1];
```

```
   T2b(:,:,i) = [cos(bet(i)) 0 sin(bet(i)); 0 1 0; -
sin(bet(i)) 0 cos(bet(i))];
   T3p(:,:,i) = [cos(psi(i)) sin(psi(i)) 0; -sin(psi(i))
cos(psi(i)) 0; 0 0 1];
end

for i = 1:k
   m3(:,:,i) = (T3z(:,:,i)*T2b(:,:,i)).';
   m4(:,:,i) =
(T3z(:,:,i)*T2b(:,:,i)*T3p(:,:,i)*T1r*T2r).';
end

syms h e1 R mb1 mb2 mb3 mb4 mb5 mb6 mb7

tempmb = [mb1 mb2 mb3 mb4 mb5 mb6 mb7];
mb = tempmb(1:k);
u1 = sym('u1(t)');
u2 = sym('u2(t)');

rhoFI_I = [u1 u2 0];
rhoHF = [0 0 h];
rhoBuH =  [e1 0 0];  % changed from vector form
rhoPBd = [R 0 0];
rhoHF_I = (m1*rhoHF.').';

for i = 1:k
   rhoBuH_I(:,:,i) = m1*m2(:,:,i)*rhoBuH.';
   rhoPBd_I(:,:,i) = m1*m2(:,:,i)*m3(:,:,i)*rhoPBd.';
   rho(:,:,i) =
(rhoFI_I+rhoHF_I+rhoBuH_I(:,:,i)+rhoPBd_I(:,:,i)).';
   V(:,:,i) = diff(rho(:,:,i),t);
   Vsqr(:,:,i) = V(1,:,i)^2 + V(2,:,i)^2 + V(3,:,i)^3;
   TBk(:,:,i) = 1/2*mb(i)*Vsqr(:,:,i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Potential Energy of kth blade (UBk)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

syms Ke1 Ke2 Ke3 Kd1 Kd2 Kd3 Ks1 Ks2 Ks3 z
syms kf11 kf12 kf13 kf14 kf15 kf16 kf17 kf31 kf32 kf33 kf34
kf35 kf36 kf37
```

```
syms kl11 kl12 kl13 kl14 kl15 kl16 kl17 kl31 kl32 kl33 kl34
kl35 kl36 kl37
syms Ke4 Ke5 Ke6 Ke7 Kd4 Kd5 Kd6 Kd7 Ks4 Ks5 Ks6 Ks7
Ket = [Ke1 Ke2 Ke3 Ke4 Ke5 Ke6 Ke7];
Ke = Ket(1:k);
Kdt = [Kd1 Kd2 Kd3 Kd4 Kd5 Kd6 Kd7];
Kd = Kdt(1:k);
Kst = [Ks1 Ks2 Ks3 Ks4 Ks5 Ks6 Ks7];
Ks = Kst(1:k);
kf1t = [kf11 kf12 kf13 kf14 kf15 kf16 kf17];
kf1 = kf1t(1:k);
kf3t = [kf31 kf32 kf33 kf34 kf35 kf36 kf37];
kf3 = kf3t(1:k);
kl1t = [kl11 kl12 kl13 kl14 kl15 kl16 kl17];
kl1 = kl1t(1:3);
kl3t = [kl31 kl32 kl33 kl34 kl35 kl36 kl37];
kl3 = kl3t(1:3);

for i = 1:3
   UBk1 = 1/2*(bet(i)^2*kf1(i)+zet(i)^2*kl1(i));
   UBk3 = 1/4*(bet(i)^4*kf3(i)+zet(i)^4*kl3(i));
   UBk(i) = UBk1 + UBk3; % + UBk3; not required for simple
model
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Dissapative Energy
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

syms DB1 DB2 DB3 Czeta1 Czeta2 Czeta3 Vzeta1 Vzeta2 Vzeta3
syms cf1 cf2 cf3 cf4 cf5 cf6 cf7
syms cl1 cl2 cl3 cl4 cl5 cl6 cl7
cft = [cf1 cf2 cf3 cf4 cf5 cf6 cf7];
cf = cft(1:k);
clt = [cl1 cl2 cl3 cl4 cl5 cl6 cl7];
cl = clt(1:3);
Czeta = [Czeta1 Czeta2 Czeta3]; % convert to 7 blades
matrix
Vzeta = [Vzeta1 Vzeta2 Vzeta3];

for i = 1:3
   DBk(i) = 1/2*diff(bet(i),t)^2*cf(i) +
1/2*diff(zet(i),t)^2*cl(i);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Energy Expression for Fuselage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Kinetic Energy of Fuselage (TF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
syms K1 K2 c1 c2 v1 v2 M1 M2 I11 I22 I12
c = [c1 c2];
v = [v1 v2];
M = [M1 M2];
K = [K1 K2];

TFt = 1/2*(diff(u1,t)^2*M(1)+diff(u2,t)^2*M(2));
TFr = 1/2*(diff(r1,t)^2*I11 + diff(r2,t)^2*I22-
2*diff(r1,t)*diff(r2,t)*I12);
TF = TFt + TFr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Potential Energy of Fuselage
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
syms KT1 KT2 KR1 KR2

UFt = 1/2*u1^2*KT1 + 1/2*u2^2*KT2;
UFr = 1/2*r1^2*KR1 + 1/2*r2^2*KR2;
UF = UFt + UFr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Dissapation Energy of Fuselage (DF)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
syms CT1 CT2 CR1 CR2 VT1 VT2 VR1 VR2

DFtv = 1/2*diff(u1,t)^2*CT1 + 1/2*diff(u2,t)^2*CT2;
DFrv = 1/2*diff(r1,t)^2*CR1 + 1/2*diff(r2,t)^2*CR2;
```

```
DFth =
1/2*diff(u1,t)^2*abs(diff(u1,t))*VT1+1/2*diff(u2,t)^2*abs(d
iff(u2,t))*VT2;
DFrh =
1/2*diff(r1,t)^2*abs(diff(r1,t))*VR1+1/2*diff(r2,t)^2*abs(d
iff(r2,t))*VR2;
DF = DFtv + DFrv + DFth + DFrh;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Aerodynamic (Generalized Aerodynamic Forces)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
syms thet1 thet2 thet3 thet4 thet5 thet6 thet7 cd0 mu
lambda rho1 a c
thett = [thet1 thet2 thet3 thet4 thet5 thet6 thet7];
thet = thett(1:k);  %  do for 7 blades

Vair = [mu*Omega*R, 0, Omega*lambda*R];
for i = 1:k
   V_I_t(:,:,i) = (-V(:,:,i)+Vair.').';
   V_Bd_t(:,:,i) = simplify(m4(:,:,i)*V_I_t(:,:,i)).';
end

for i = 1:k
   UR(:,:,i) = V_Bd_t(1,:,i);
   UT(:,:,i) = -V_Bd_t(2,:,i);
   UP(:,:,i) = -V_Bd_t(3,:,i);
   UU(:,:,i) = sqrt(UP(:,:,i)^2+UT(:,:,i)^2);
end

for i = 1:k
   aoa(:,:,i) = thet(i)-atan(UP(:,:,i)/UT(:,:,i));
   dFbeta(:,:,i) =
1/2*rho1*a*c*(aoa(:,:,i)*UU(:,:,i)*UT(:,:,i)-
cd0/a*UU(:,:,i)*UP(:,:,i));
   dFzeta(:,:,i) = -
1/2*rho1*a*c*(aoa(i)*UU(:,:,i)*UP(:,:,i)+cd0/a*UU(:,:,i)*UT
(:,:,i));
   Mbeta_k(:,:,i) = 0.7*R^2*dFbeta(:,:,i)*cos(zet(i));
   Mzeta(:,:,i) = 0.7*R^2*dFzeta(:,:,i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
```

```
% Derivation of Equations of Motion by Lagrangian Method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
syms

DOFF = [u1, u2, r1, r2];
DOFB = [];
DOFB = [bet,zet];

DOF = [DOFF, DOFB];
dDOF = diff(DOF,t);
ddDOF = diff(dDOF,t);
[1,n] = size(DOF);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
% Transformation between time dependent and independent
notation by substituting sets
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%
syms q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 dq1 dq2 dq3 dq4 dq5 dq6
dq7 dq8 dq9 dq10
syms ddq1 ddq2 ddq3 ddq4 ddq5 ddq6 ddq7 ddq8 ddq9 ddq10
syms q11 q12 q13 q14 q15 q16 q17 q18 dq11 dq12 dq13 dq14
dq15 dq16 dq17 dq18
syms ddq11 ddq12 ddq13 ddq14 ddq15 ddq16 ddq17 ddq18

DOFqt = [q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 q12 q13 q14 q15
q16 q17 q18];
DOFq = DOFqt(1:n);
dDOFqt = [dq1 dq2 dq3 dq4 dq5 dq6 dq7 dq8 dq9 dq10 dq11
dq12 dq13 dq14 dq15 dq16 dq17 dq18];
dDOFq = dDOFqt(1:n);
ddDOFqt = [ddq1 ddq2 ddq3 ddq4 ddq5 ddq6 ddq7 ddq8 ddq9
ddq10 ddq11 ddq12 ddq13 ddq14 ddq15 ddq16 ddq17 ddq18];
ddDOFq = ddDOFqt(1:n);

set1 = [];
set2 = [];
for i=1:n       % size of DOF vector
   set1 = [set1 maple('='',DOF(i),DOFq(i)) ];
   set1 = [set1 maple('='',dDOF(i),dDOFq(i)) ];
   set1 = [set1 maple('='',ddDOF(i),ddDOFq(i)) ];
   set2 = [set2 maple('='',DOFq(i),DOF(i)) ];
   set2 = [set2 maple('='',dDOFq(i),dDOF(i)) ];
```

```
   set2 = [set2 maple('`=`',ddDOFq(i),ddDOF(i)) ];
end

set1 = maple('convert',set1,'set');
set2 = maple('convert',set2,'set');

T = TF;
U = UF;
D1 = DF;
GF = [0,0,0,0];

[l,n] = size(DOFB);
for i = 1:k
   T = T+TBk(:,:,i);
   U = U+UBk(i);
   D1 = D1+DBk(i);
   GF = [GF Mbeta_k(:,:,i) Mzeta(:,:,i)];
end

Temp = maple('subs',set1,T);  %Temp = subs(T,dDOF,dDOFq);
save for time comparison
GFq = maple('subs',set1,GF);  %GFq = subs(GF,dDOF,dDOFq);
save for time comparison

[l,n] = size(DOF);
for i = 1:n
   temp1 = diff(Temp,dDOFq(i));
   temp2 = maple('subs',set2,temp1);
   temp3 = diff(temp2,t);
   L1 = maple('subs',set1,temp3);
   L2 = diff(Temp,DOFq(i));
   tempL3 = maple('subs',set1,U);
   L3 = diff(tempL3,DOFq(i)); % check dimensions of DOFq
   tempL4 = maple('subs',set1,D1);
   L4 = diff(tempL4,dDOFq(i)); % check dimension of DOFq
   %  EOM(i) = simplify(L1-L2+L3+L4-GFq(i));
   EOM(i) = (L1-L2+L3+L4-GFq(i));  % input to allow
computation, gags on simplify(EOM)
end


% code will generate EOM, will not perform operations on
EOM, ie. simplify, eval, etc.
% remainder of code should be correct, could not evaluate
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% Format equations of motion into form  A d2x/dt2 = f
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
for i = 1:n
   for j = 1:n
      A(i,j) = maple('coeff',EOM(i),ddDOFq(j));
   end
end

[j,g] = size(ddDOFq);
setZ = zeros(1,g);

f(1:g) = -eval(subs(EOM(1:g),setZ,ddDOFq));

Ax2dot = ddDOFq*A;
for i = 1:n
   f(i) = (EOM(i)-Ax2dot(i));
   f(i) = maple('subs',setS,f(i));
   f(i) = -(simplify(f(i)));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% Change notation for input into S-function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16
x17 x18

X1temp = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14
x15 x16 x17 x18];
X1dot = X1temp(1:n);
X1 = X1temp(n+1:2*n);
X1 = [X1temp X1dot];
DOFqtemp = [DOFq dDOFq];

setX = [];
for i=1:2*n
   setX = [setX maple('=',DOFqtemp(i),X1(i)) ];
end
setX = maple('convert',setX,'set');

tempsetX1 = [abs(1,x1) abs(1,x2) abs(1,x3) abs(1,x4)];
```

```
tempzeros = zeros(size(tempsetX1));

A1 = maple('subs',setX,A);
f1 = maple('subs',setX,f);
f2 = subs(f1,tempsetX1,tempzeros);

syms u1 u2 u3
ut = [u1 u2 u3];
A1 = subs(A1,u,ut);
f2 = subs(f2,u,ut);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. MATLAB® FUNCTION TO ACCESS THE MAPLE FUNCTION 'ABS' TO CALCULATE THE ABSOLUTE VALUE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% abs.m
% This function is designed to call the Maple function
% 'abs' with two inpust, b and c and return the result
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function a = abs(b,c)
a = maple('abs',b,c);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. MATLAB® FUNCTION TO ACCESS THE MAPLE FUNCTION 'SIGNUM'

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% signum.m
% This function is designed to call the Maple function
% 'signum' evaluating one expression, a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function s = signum(a)
s = maple('signum',a);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. MATLAB® FUNCTION TO ACCESS THE MAPLE FUNCTION 'SIGNUM', TAKING THE DERIVATIVE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%signum1.m
% This function is design to call the Maple function
% 'signum' using a variation of the function with
% a = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function s = signum(a,b)
s = maple('signum',a,b);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. MATLAB® FUNCTION MUNION, TAKING THE UNION OF TWO SETS USING THE MAPLE FUNCTION

```
function sym_union = munion(arg1,arg2)

stringa = char(arg1);  %get rid of extra char's
stringa = stringa(8:end-2);

if length(stringa) == 0
   stringa = '{}';
end

stringb = char(arg2);
stringb = stringb(8:end-2);

if length(stringb) == 0
   stringb = '{}';
end


union_string = [stringa,' union ',stringb];

union_string(union_string == ']') = '}';
union_string(union_string == '[') = '{';


union_string = ['maple('' ', union_string, ' '')'];

temp1 = sym(eval(union_string));

temp2 = char(temp1); %[temp1(temp1 == '{') = '[']

temp2(temp2 == '{') = '[';
temp2(temp2 == '}') = ']';

sym_union = sym(temp2);
```

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code AA/Co 1
   Department of Aeronautics and Astronautics
   411 Dyer Rd.
   Monterey, CA 93943-5101

4. Dr. E. Roberts Wood 1
   Naval Postgraduate School
   Code AA/WD
   699 Dyer Road
   Monterey, CA 93943-5106

5. Dr. Robert L. King 1
   Department of Aerospace Engineering
   Mississippi State University
   P.O. Box A
   Mississippi State, MS 39762-5501

6. Dr. Gary Anderson 1
   U. S. Army Research Office
   P.O. Box 1211
   Research Triangle Park, NC 27709

7. Robert D. Weissenfels 3
   1943 Harris St.
   Richland, WA 99352

8. Dr. Robert Ormiston 1
   Building: 215, Room: 201C
   NASA Ames Research Center
   MS 215-1
   Moffet Field, CA 94035-1000

9. Dr. Jeffrey D. Singleton 1
Vehicle Structure Directorate
U.S. Army Research Laboratory
Mail Stop 340
NASA Langley Research Center
Hampton, VA 23681-0001

10. Mr. Robert Blackwell 1
103 Old Zoar Rd
Monroe, CT 06468

11. Dr. Friedrich Straub 1
1760 E. Halifax St
Mesa, AZ 85203